WEB- & N-TIER TECHNOLOGIES

Eine Einführung in Web-Technologie und verteilte Applikationen

Autor: Walter Rothlin (29.1.21)

HWZ Hochschule für Wirtschaft Zürich

```
10 INPUT "Geben Sie bitte Ihren Namen ein"; A$
20 PRINT "Guten Tag, "; A$
30 INPUT "Wie viele Sterne möchten
35 S$ = ""
40 FOR I = 1 TO S
50 S$ = S$ + "*"
55 NEXT I
60 PRINT S$
70 INPUT "Möchten Sie noch mehr St
80 IF LEN(Q$) = 0 THEN GOTO 70
90 L$ = LEFT$(Q$, 1)
100 IF (L$ = "J") OR (L$ = "j") THEN GOTO 30
110 PRINT "Auf Wiedersehen";
120 FOR I = 1 TO 200
130 PRINT A$; " ";
140 NEXT I
150 PRINT
```

<u>Sequenzen, if-goto</u>

- Each application starts from scratch
- Maintenance intensive
- Migrationtests
- Reuse via cut / past
- Predefined functions as part of the language

```python
# ------------------------------------------------------------------
# Name: umrechnungen.py
#
# Description: Rechnet verschiedene physikalische Grössen um.
#
# Autor: Walter Rothlin
#
# History:
# 26-Sep-2017   Walter Rothlin      Initial Version
#
# ------------------------------------------------------------------
doLoop = True
while doLoop:
    print("  Umrechnungen")
    print("  ============")
    print("  1: Grad in Bogenmass")  # rad  = grad*pi/180
    print("  2: Bogenmass in Grad")  # grad = rad*180/pi
    print()
    print("  3: Fahrenheit in Celsius")  #32F -> 0°C    100F -> 38.8°C     °C = (°F - 32) / 1.8
    print("  4: Celsius in Fahrenheit")  #32F -> 0°C    100F -> 38.8°C     °F = (°C * 1.8) - 32
    print()
    print("  0: Schluss")


    antwort = input("\n  Wähle:")
    if (antwort == "1"):
        gradValue=float(input("Grad:"))
        print(gradValue, "Grad ==> ", gradValue*3.141592/180, "Rad")

    if (antwort == "2"):
        radValue=float(input("Rad:"))
        print(gradValue, "Rad ==> ", radValue*180/3.141592, "Grad")

    if (antwort == "3"):

    if (antwort == "4"):

    if (antwort == "0"):
        doLoop = False

print("Ende....Done")
```

Loops and If-Then-Else

+ Better Maintenance

HWZ Hochschule für Wirtschaft Zürich

```python
# ------------------------------------------------------------------
def grad2Rad(grad):
    return 3.1415*grad/180

def rad2Grad(rad):
    return 180*rad/3.1415

def fahrenheit2Celsius(fahrenheit):
    return (fahrenheit-32)/1.8

def celsius2Fahrenheit(celsius):
    return (celsius*1.8)+32

doLoop = True
while doLoop:
    VT52_cls_home()
    print("  Umrechnungen")
    print("  ============")
    print("  1: Grad in Bogenmass")  # rad  = grad*pi/180
    print("  2: Bogenmass in Grad")  # grad = rad*180/pi
    print()
    print("  3: Fahrenheit in Celsius")  #32F -> 0°C    100F -> 38.8°C     °C = (°F - 32) / 1.8
    print("  4: Celsius in Fahrenheit")  #32F -> 0°C    100F -> 38.8°C     °F = (°C * 1.8) - 32
    print()
    print("  0: Schluss")


    antwort = input("\n  Wähle:")
    if (antwort == "1"):
        VT52_cls_home()
        print("Grad --> Bogenmass")
        gradValue=float(input("Grad:"))
        print("Grad={grad:1.2f}  ==> Rad={rad:1.2f}".format(grad=gradValue,rad=grad2Rad(gradValue)))
        halt()
```

Own functions with parameters

+ Easier maintenance

+ Elimination of redundant code

HWZ | Hochschule für Wirtschaft Zürich

```python
from waltisLibrary import *

doLoop = True
while doLoop:
    VT52_cls_home()
    print("  Umrechnungen")
    print("  ============")
    print("  1: Grad in Bogenmass")  # rad  = grad*pi/180
    print("  2: Bogenmass in Grad")  # grad = rad*180/pi
    print()
    print("  3: Fahrenheit in Celsius")  #32F -> 0°C    1
    print("  4: Celsius in Fahrenheit")  #32F -> 0°C    1
    print()
    print("  0: Schluss")


    antwort = input("\n  Wähle:")
    if (antwort == "1"):
        VT52_cls_home()
        print("Grad --> Bogenmass")
        gradValue=float(input("Grad:"))
        print("Grad={grad:1.2f}  ==> Rad={rad:1.2f}".format(grad=gradValue,rad=grad2Rad(gradValue)))
        halt()

    if (antwort == "2"):
        VT52_cls_home()
        print("Bogenmass --> Grad")
        radValue=float(input("Rad:"))
        print("Rad={rad:1.2f}  ==> Grad={grad:1.2f}".format(rad=radValue,grad=rad2Grad(radValue)))
        halt()

    if (antwort == "3"):
        VT52_cls_home()    # http://www.metric-conversions.org/de/temperatur/fahrenheit-in-celsius.htm
        print("Fahrenheit in Celsius")
        fahrenheitValue=float(input("Fahrenheit:"))
```

Function Libraries

+ Easier Maintenance
+ Reuse possible
+ Elimination of redundant code
+ Separation application / reusable libs

```python
# ---------------------------------------------
import math
import os
import sys
import time
import datetime

from time       import sleep

# Primzahlen Functions
# --------------------
def isPrimzahl(aZahl):
    isPrim = True
    if (aZahl == 1):
        isPrim = True
    else:
        if (aZahl == 2):
            isPrim = True
        else:
            obergrenze = int((aZahl / 2) + 2)
            for i in range(2,obergrenze):
                if ((aZahl % i) == 0):
                    isPrim = False
    return isPrim

def getNextPrimzahl(zahl):
    aZahl = zahl + 1
    while (isPrimzahl(aZahl) == False):
        aZahl = aZahl + 1
    return aZahl
```

External Function Libraries

+ Reuse possible
+ Because of Internet
    + open source
    + sharing for free
+ IDE simplifies to import
- No states
- Language dependency

```
if (inTestMode):
    print("IN TEST-MODE")
    print("   time_T2                    : ",time_T2)
    print("   schwellwert_H_OffImmediate: ",schwellwert_H_OffImmediate)
    print()
    print("=====================\n")
print(hsrKaelteMaschine.toString())
logFile.addLogEntry(hsrKaelteMaschine.toStringForLog())
if (hsrKaelteMaschine.isEmergencyOff_Active()):
    print("--------------------------------------1")
    print("Emergency Off pressed!! Will not start!!!")
    hsrKaelteMaschine.doEmergencyStop()
    verdichter_1.doEmergencyStop()
    verdichter_2.doEmergencyStop()
    # t2State.setState_NotStarted() # Waterpump steys on in EmercencyOff
else:
    print("--------------------------------------2")
    vorhandeneEnergie           = hsrKaelteMaschine.getExistingEnergy()
    hochdruck                   = hsrKaelteMaschine.getHighPressure()
    niederdruck                 = hsrKaelteMaschine.getLowPressure()
    waterTemp                   = hsrKaelteMaschine.getWaterTemp_PT1000()
    envTemp                     = hsrKaelteMaschine.getEnvironmentTemp_PT1000()

    # ---------------------------------------------
    # Calculated values
    # ---------------------------------------------
    lowPressureError            = niederDruckOff.setState(niederdruck,verbal = True)
    highPressureError           = hochDruckOff.setState(hochdruck    ,verbal = True)
    pressureError = (lowPressureError or highPressureError)
    if (pressureError):
        if (t1State.isState_NotStarted()):
            t1 = Timer(time_T1, T1Over, args=[t1State])
            t1State.setState_Ticking()
            t1.start()
        print("pressureError       :",pressureError,end="")
        if (lowPressureError):
           print("  (lowPressureError)",end="")
           hsrKaelteMaschine.setShutOffValveToOpen(True)
           time.sleep(lowPressureErrorShutOffValveReopenTime)
```

Classes: Methods with states

+ Reuse improved (inheritance)

+ Decoppeling

+ Simple interfaces

- Language dependend

- No runtime distribution

```python
# ------------------------------------------------------------------
import math
import time
import datetime

from waltisLibrary import *

class Verdichter:

    # Ctr (Konstruktor)
    # -----------------
    def __init__(self, name, pMin, pMax, outputTiefpass, hsrKaelteMaschine):
        self.name                       = name
        self.pMin                       = pMin
        self.pMax                       = pMax
        self.outputTiefpass             = outputTiefpass
        self.hsrKaelteMaschine          = hsrKaelteMaschine
        self.istSpeed                   = 0
        self.sollSpeed                  = 0
        self.verdichterOn               = False
        self.isVerdichterOnState        = False
        self.wiederEinschaltenMoeglich  = False
        self.lastTimeOff                = datetime.datetime.now()

    # Methoden (setter / getter)
    # --------------------------
    def setValues(self, sollSpeed, emergencyOff, einschaltverzoegert):
        currTime = datetime.datetime.now()
        wiedereinschaltverzoegerung = "0:00:10"
        print("==> setValues ({x1:1s})::sollSpeed:{x2:4.0f}%".format(x1=self.name,x2

        if (emergencyOff):
            self.sollSpeed    = 0
            self.verdichterOn = False
            self.resetRestartTime()
        else:
            if (sollSpeed > self.pMax):
                self.sollSpeed = self.pMax
                self.verdichterOn = True
            elif (sollSpeed < self.pMin):
                self.sollSpeed = 0
                if (self.verdichterOn):
                    self.verdichterOn = False
```

Class-Libraries (Packages)

+ Reuse improved
+ Decoppeling
+ Simple interfaces
- Language dependend
- No runtime distribution

```java
package com.rothlin.hwz;

import java.io.IOException;


@WebServlet("/HWZ_SimpleEJB_Servlet_1")
public class HWZ_SimpleEJB_Servlet_1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @EJB
    private TestHWZ_EJB testEJB;

    public HWZ_SimpleEJB_Servlet_1() {
        super();
    }

    private void serviceRequest(
            HttpServletRequest  request,
            HttpServletResponse response, boolean callFrom_doGet)
            throws ServletException, IOException {
        StringBuffer sb      = new StringBuffer();
        sb.append("<html>\n<head>\n</head>\n<body>\n");
        sb.append("Hallo!!!!" + testEJB.sayHello());
        sb.append("</body>\n</html>\n");
        response.setContentType("text/html");
        response.getWriter().print(sb);
    }
```

Distributed Objects

+ Component reuse
+ Scallable
+ Simple interfaces
- Language dependend

```java
package com.rothlin.hwz;
import javax.ejb.LocalBean;

@Stateless
@LocalBean
public class TestHWZ_EJB {


    public TestHWZ_EJB() {}

    public String sayHello() {
        return "Hallo, wo gibt es ein Bier?";
    }

}
```

Distributed Objects

+ POJO

- Language dependend

```java
private FirstWebServiceProxy ws;

public WebServiceClientOO() {
    ws = new FirstWebServiceProxy();
}

public static void main(String[] args) {
    try {
        WebServiceClientOO client = new WebServiceClientOO();
        client.doTest(args);
    } catch(Exception e) {
        e.printStackTrace();
    }
}

public void doTest(String[] args) {
    try {
        System.out.println(ws.ping());
        System.out.println(ws.sayHelloName("BWI-15"));
        System.out.println(ws.sayAnredeName("Herr","Peter"));
        System.out.println(ws.calcCircleArea(5.0));

    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

Web-Services

+ Simple

+ Simple

+ Language independent

```java
@WebService
public class FirstWebService {

    // Methodes in a web-Service are called Operations
    // Operation names can not be overloaded!!
    @WebMethod(exclude=false)
    public String ping() {
        return "Webservice (1.0.0.0)::FirstWebService:ping() called!!!!!!";
    }

    @WebMethod() // ctrl-space shows you the options
    public String sayHelloName(String name) {
        return "Hello " + name;
    }

    @WebMethod() // ctrl-space shows you the options
    public String sayHelloHWZ(String name) {
        return "Hello du " + name + "! Gruss von der HWZ";
    }

    @WebMethod(exclude=false)
```

Web-Services

+ Simple

+ Simple

+ Language independent

Application / Web-Application

- Process (Programm), der auf Umgebung reagiert
  - User-Interactions
  - Via sensors (input) and actors (output)
- Web / HTML ➔ strukturierte und verlinkte Dokumente
  - Via URL
  - Page based (reload)
  - Client (Web-Browser) sanded requests an den Server (Web-Server)
  - Web-Server sendet eine Response zurück
- Web-Application läuft im Browser
  - Eingabe Felder (Form-Elemente) / Layout (CSS)
  - Abhängig von Events müssen Actions ausgelöst werden ➔ Programmlogik durch JavaScript (JS) im HTML-Document
  - Actions können durch JS dynamisch (ohne reload) DOM verändern
  - Actions können auch Request auslösen und die Response wird von JS im DOM verarbeitet

Examples Web-Application

- Adrino: http://192.168.1.20/
- SBB Fahrplan: https://www.sbb.ch/
- Google search: https://www.google.ch/
- URL verkürzen (Bitly): https://bitly.com/
- QR-Code generator: https://www.qrcode-generator.de
  - Web-Service: http://goqr.me/de/api/doc/
  - Beispiel: http://api.qrserver.com/v1/create-qr-code/?data=HelloWorld!&size=100x100
- Map Geo: https://map.geo.admin.ch
  - API: http://api.geo.admin.ch/
  - REST: https://api3.geo.admin.ch/1912100956/rest/services/ech/SearchServer?sr=2056&searchText=peterliwiese%2033&lang=en&type=locations
  - Application: http://www.fhoch3.peterliwiese.ch/AdresseToKoordinates.html
  - Excel calls REST: https://www.geo.admin.ch/de/news/aktuell.detail.news.html/geo-internet/news2019/news20190307.html
  - Konsultieren Sie bitte hierzu api.geo.admin.ch Beispiel zur Nutzung des RESTService zb in dieser excel formel https://www.geo.admin.ch/de/news/aktuell.detail.news.html/geo-internet/news2019/news20190307.html
  - Das API von geo.admin.ch darf auf Webseiten eingesetzt werden, sofern diese Verwendung den Nutzungsbedingungen von GeoAdmin und CadastralWebMap entsprechen. Die Registrierung und Nutzungsbedingungen finden Sie unter: https://www.geo.admin.ch/de/geo-dienstleistungen/geodienste/darstellungsdienste-webmapping-webgis-anwendungen/programmierschnittstelle-api/anmeldung.html
- Open Weather: https://openweathermap.org/
  - API: https://openweathermap.org/api
  - REST: http://api.openweathermap.org/data/2.5/weather?q=Wangen%20SZ&appid=3836093dde650898eb014e6f27304646
  - Token: **3836093dde650898eb014e6f27304646 walter rothlin 8....n**

HWZ Hochschule für Wirtschaft Zürich

Ziel

Die Studierenden

- erkennen und beschreiben Probleme der Web- resp. Verteilter Software-Entwicklung
- realisieren einfache Lösungen mit J2EE Technologien
- beschreiben die Grundlagen mehrschichtiger Software-Anwendungen am Beispiel von J2EE
- erstellen einer einfache, mehrschichtige Anwendung mit Eclipse
- greifen von Java-Programmen auf eine SQL-Datenbank zu

URL Uniform Resource Locator

Eine URL ist eine weltweit eindeutige Adresse er Resource (z.B. Datei)

z.B. http://www.fh-hwz.ch:80/Download/b1a/Semesterprogramm.pdf

`http:`	Protokoll (https)	Wie?
`www.fhhwz.ch:`	Full qualified host name	Mit wem?
`80:`	Port-Nummer	
`Download/b1a:`	Directory-Struktur	Was?
`Semesterprogramm.pdf:`	Document name	

Protokoll

Definiert wie zwei Kommunikations-Partner miteinander sprechen.

Im World Wide Web (www)

`http`	Hyper text transfer protokoll
`https`	http-Secure

Weitere Internet Protokolle

`ping`	Verbindung testen	
`telnet`	remote Console	
`ftp`	File transfer	
`sftp`	secure file transfer	(Public-Private Key)
`ftps`	secure file transfer	(Certificates)
`smtp`	simple mail tranfer protokol	

Host- und Domain-Names

Jeder Computer in einem Netzwerk muss eindeutig bezeichnet sein!

➔ IP Address:

- Eindeutige Nummer (4-stellig, jede Stelle 0..255: 255^4=4Mia)
- z.B. 107.18.128.198

Nummern kann man sich schlecht merken! Namen bringen mehr Flexibilität!

➔ Hostname / Domain-Name:

- Eindeutiger Name für einen Computer
- z.B. celera.credit-suisse.ch

2nd Level Domain

1st Level Domain

Domain-Name
Hostname = Full qualified hostname

Übersetzt IP-Adressen in Hostnames und umgekehrt: ➔DNS (Domain Name Service)

`ipconfig /all`	zeigt die eigene IP Adresse
`nslookup`	übersetzt IP ← → Hostname

Port

Auf einem Computer sind mehrere Zuhörer / Sprecher gleichzeitig aktiv. Der Port bestimmt mit welchem ich sprechen will.

Dienste	Port	Protokoll
echo	7	TCP
systat	11	TCP
chargen	19	TCP
ftp-data	21	TCP
ssh	22	TCP
telnet	23	TCP
smtp	25	TCP
nameserver	42	TCP
whois	43	TCP
tacacs	49	UDP
dns-lookup	53	UDP
dns-zone	53	TCP
oracl-sqlnet	66	TCP
tftp	69	UDP
finger	79	TCP
http	80	TCP
http-1	81	TCP
kerberos	88	TCP
pop2	109	TCP
pop3	110	TCP
sunrpc	111	TCP
sqlserv	118	TCP

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

0 to 1023
are the *well-known ports* or *system ports*

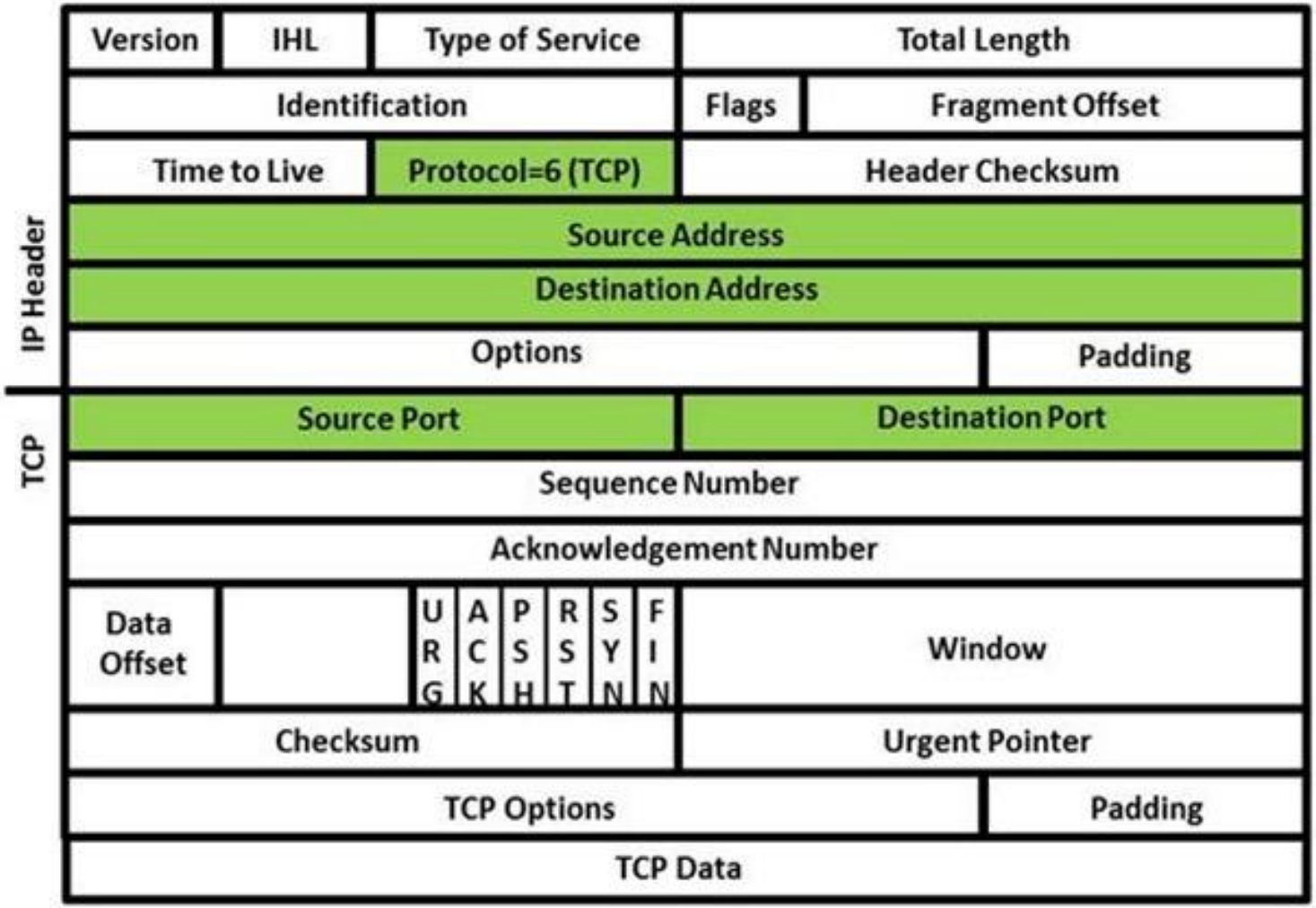
1024 to 49151
are the registered ports

e.g. 8080 alternate http

49152–65535 ($2^{15}+2^{14}$ to $2^{16}-1$)
contains dynamic or private ports

Transportation

TCP/IP Packet



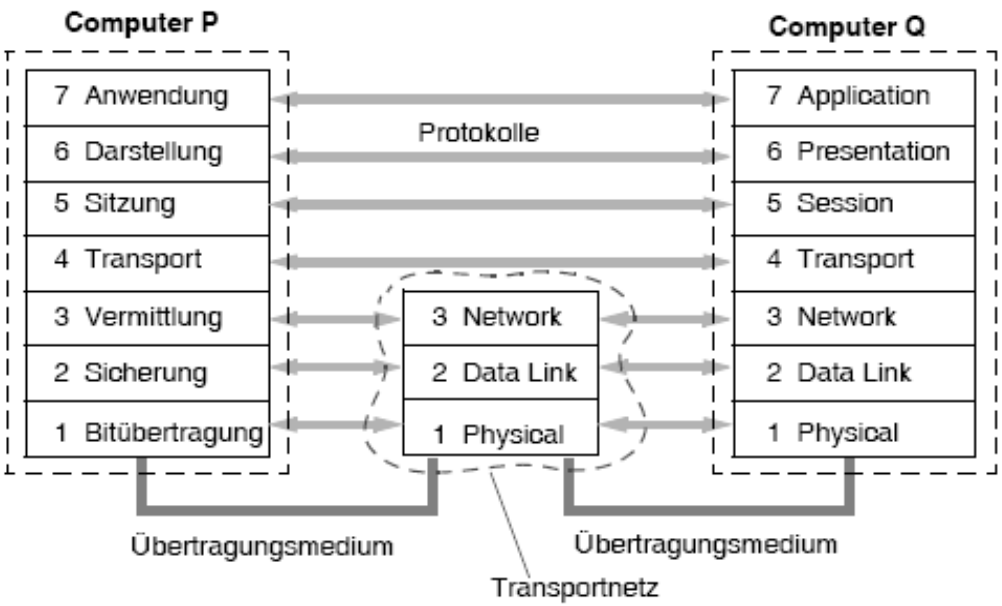
32-Bit / Zeile

IP-Adresse

z.B. 158.2.34.150

Port Nr: 0.. 65535

TCP/IP *Open Systems Interconnect (OSI) Reference Model*



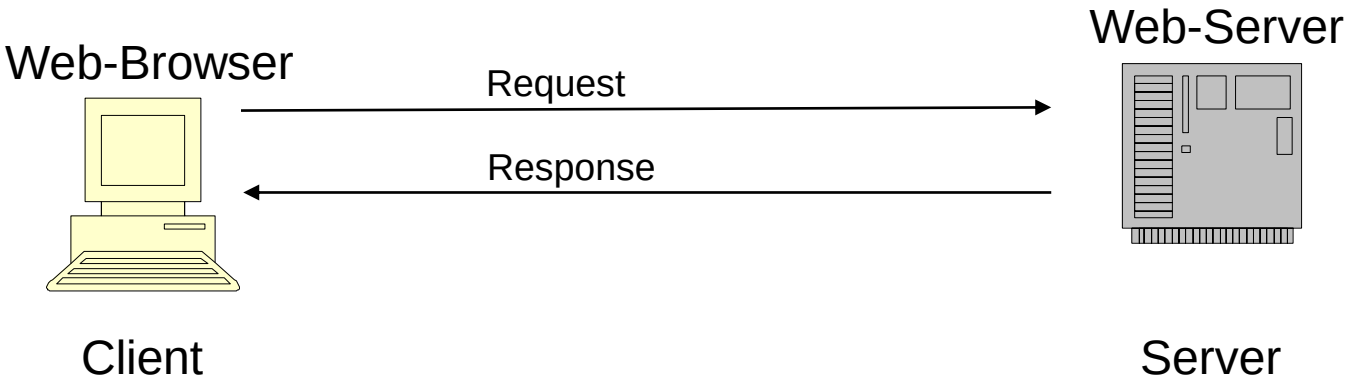
7	**Anwendungsschicht** besteht aus den Anwendungen mit denen man das Netz nutzen kann
6	**Darstellungsschicht** standadisiert das Format der Daten auf dem Netz
5	**Kommunikationssteuerungsschicht** verwaltet die Verbindungen zwischen den Anwendungen
4	**Transportschicht** garantiert die fehlerfreie Datenübertragung durch Fehlererkennung und -korrektur
3	**Vermittlungsschicht** verwaltet die Verbindungen zwischen den Rechnern im Netz für die höheren Schichten
2	**Sicherungsschicht** sorgt für die zuverlässige Übertragung der Daten über die physikalischen Verbindungen
1	**Bitübertragungsschicht** definiert die physikalischen Eigenschaften der Übertragungswege

Web-Server

Auf einem Server (Computer mit IP Adresse eindeutig in einem Netzwerk bestimmt) ist auf einem (Software-)Port (Default 80) ein Prozess am laufen, welcher http versteht ➔ Web-Server

- ➔ Document-Root: Legt oberstes Verzeichnis des Web-Servers im file-System fest
- ➔ Mime-Type Table: Zuordnung von File-Extensions zu Mime-Types. Wird vor dem zurück-streamen des Content (File-Inhalt der URL)
- ➔ Unterscheidungsmerkmal zwischen static und dynamic-Content
- ➔ Interpretiert URL in http request und generiert eine respond
- ➔ Sobald die Antwort gegeben ist, vergisst der Web-Server alles über den Request (stateless)

Zusammenfassung:



1. Web-Server muss laufen; Web-Browser starten
2. URL im Browser eintippen und mit Return abschliessen
3. Request wird abgesendet und sucht sich den Weg zum Web-Server (via Internet, IP-Adresse und Port)
4. Web-Server interpretiert Request und gibt eine Response (mit Mime-Type) zurück
5. Browser rendert Antwort

Static Web Pages: HTML (Hypertext Markup Language)

- Text-Datei! Kann mit normalem Text-Editor erstellt werden (z.B. Notepad)
- Darstellung (Layout) wird vom Empfänger definiert (Rendern) und ist somit für Publisher undefiniert.
- Definiert mit Hilfe von Tags die Struktur des Dokuments
- http://www.peterliwiese.ch/AppletLib/tips/tips.html

Hello World Beispiel

```
<HTML>
  <HEAD>
    <TITLE>First example</TITLE>
  </HEAD>
  <BODY bgcolor="#FFFFFF">
  <H1>Hallo Besucher!</H1>
  Dein Besuch freut mich
  </BODY>
</HTML>
```

Start-Tag

Attribute

Value

End-Tag

Hilfe und Erklärungen auf http://de.selfhtml.org/

Cascaded Style Sheet (CSS)

HTML ist eine Markup-Language.

➔ Definiert nur Struktur des Dokuments (nicht Layout)

➔ Mit CSS können Elemente genau positioniert oder formatiert werden

Einbinden von CSS

```
<head>

 <title>Das style-Attribut</title>

 <link rel="stylesheet"                          href="formate.css">
 <link rel="stylesheet" media="screen"           href="website.css">
 <link rel="stylesheet" media="print, embossed" href="druck.css">
 <link rel="stylesheet" media="aural"            href="speaker.css">

 <style type="text/css">

 /* ... hier sind dateispezifische Formate erlaubt ... */

 </style>

 </head>
```

http://de.wikipedia.org/wiki/Cascading_Style_Sheets

http://de.selfhtml.org/css/index.htm

Cascaded Style Sheet (CSS)

Beispiel eines CSS für HTML-Elemente

```
<style type="text/css">
  body { background-color:#FFFFCC; margin-left:100px; }
  h1 { font-size:300%; color:#FF0000; font-style:italic; border-bottom:solid thin black; }
  Gilt fuer diese Elemente. Unterelemente

  * { color:blue; }
  Universalselektor; gilt fuer alle Elemente

  p,li { font-size:110%; line-height:140%; font-family:Helvetica,Arial,sans-serif; word-spacing:0.3em; }
  Eigenschaften gelten für <p> und <li>

  h1 i { color:blue; font-style:normal; }
  Eigenschaften gelten nur für <i> innerhalb von <h1>

  div * b { color:violet; }
  Eigenschaften gelten nur für <b> innerhalb (beliebig tief) von <div>
</style>
```

Cascaded Style Sheet (CSS)

Beispiel eines CSS mit attributbedingten Formaten

```
<style type="text/css">
    p[align] { color:red; }
```

Gilt für <p> bei denen das Attribute **align** gesetzt ist

```
    p[align=center] { color:blue; text-align:left; }
```

Gilt für <p> bei denen das Attribute **align** auf **center** gesetzt ist

```
    td[abbr~=Berlin] { background-color:#FFFF00 }
```

Gilt für <td> bei denen das Attribute **abbr** den Text **Berlin** enthaelt

```
   *[lang|=en] { background-color:#FF0000; color:#FFFFFF; }
```

Gilt für alle Elemente bei denen das Attribute **lang** mit dem Text **en** beginnen

```
</style>
```

Cascaded Style Sheet (CSS)

Beispiel eines CSS und deren Anwendung (class)

```
<style type="text/css">
 .beitrag { border:1px outset gray; margin:.5em; padding:.5em; background-color:#efd;}
 .uebersetzung { border-bottom:1px dotted #900; }
 .autor { font-style:italic; }
 .datum { font-size:80%; color:#444; }
 h1.hinterlegt { background-color:#FFFF00 }
 </style>
```

Unterschied und :

- Beide tags relativ eigenschaftslos
- Neue Zeile
- kein neuer Absatz (im Textfluss)

```
<div class="beitrag">
 <p>Beispielsweise in der
   <a href="http://www.w3.org/TR/REC-html40/">HTML 4.01-
   <span lang="en" title="Empfehlung" class="uebersetzung">Recommendation</span></a>.
 </p>
 <p class="autor">Max Hilfreich, <span class="datum">22.05.2006</span> </p>
 <h1 class="hinterlegt">H1 knallgelb hinterlegt</h1>
</div>
```

Cascaded Style Sheet (CSS)

Beispiel eines CSS und deren Anwendung (id)

```
<style type="text/css">
 #roterBereich {
    position:absolute; top:130px; left:30px; width:320px;
    padding:10px; margin:0px; border:4px solid #EE0000;
 }
 </style>
```

```
<div id="roterBereich"><h1>Der rote Bereich</h1></div>
```

Layout / CSS: http://www.dynamicdrive.com/style/

HTML Form-Elemente

http://www.peterliwiese.ch/AppletLib/formulare/formularElemente.html

	<FORM METHOD=POST ACTION=http://www.peterliwiese.ch/cgi-bin/waltisExamples/echo.pl>
RADIO ○ Radio 1 ◉ Radio 2 ○ Radio 3 ○ Radio 4	RADIO <INPUT TYPE=RADIO NAME=RadioButn VALUE=1>Radio 1 <INPUT TYPE=RADIO NAME=RadioButn VALUE=2 CHECKED>Radio 2 <INPUT TYPE=RADIO NAME=RadioButn VALUE=3>Radio 3 <INPUT TYPE=RADIO NAME=RadioButn VALUE=4>Radio 4
SELECT (SIZE=1) Option 2	SELECT (SIZE=1) <SELECT NAME=SelectList_1> <OPTION VALUE=1>Option 1 <OPTION VALUE=2 SELECTED>Option 2 <OPTION VALUE=3>Option 3 </SELECT>
SELECT (SIZE=2) Option 1 Option 2	SELECT (SIZE=2) <SELECT NAME=SelectList_2 SIZE=2 MULTIPLE> <OPTION VALUE=1>Option 1 <OPTION VALUE=2 SELECTED>Option 2 <OPTION VALUE=3>Option 3 <OPTION VALUE=4>Option 4 <OPTION VALUE=5>Option 5 </SELECT>
CHECKBOX ☑ Checkbox 1 ☐ Checkbox 2	CHECKBOX <INPUT TYPE=CHECKBOX NAME=Checkbox1 CHECKED> Checkbox 1 <INPUT TYPE=CHECKBOX NAME=Checkbox2>Checkbox 2
TEXT Walter_Rothlin	TEXT <INPUT TYPE=TEXT NAME=TextfieldNormal>
PASSWORD •••••	PASSWORD <INPUT TYPE=PASSWORD NAME=Passwort>

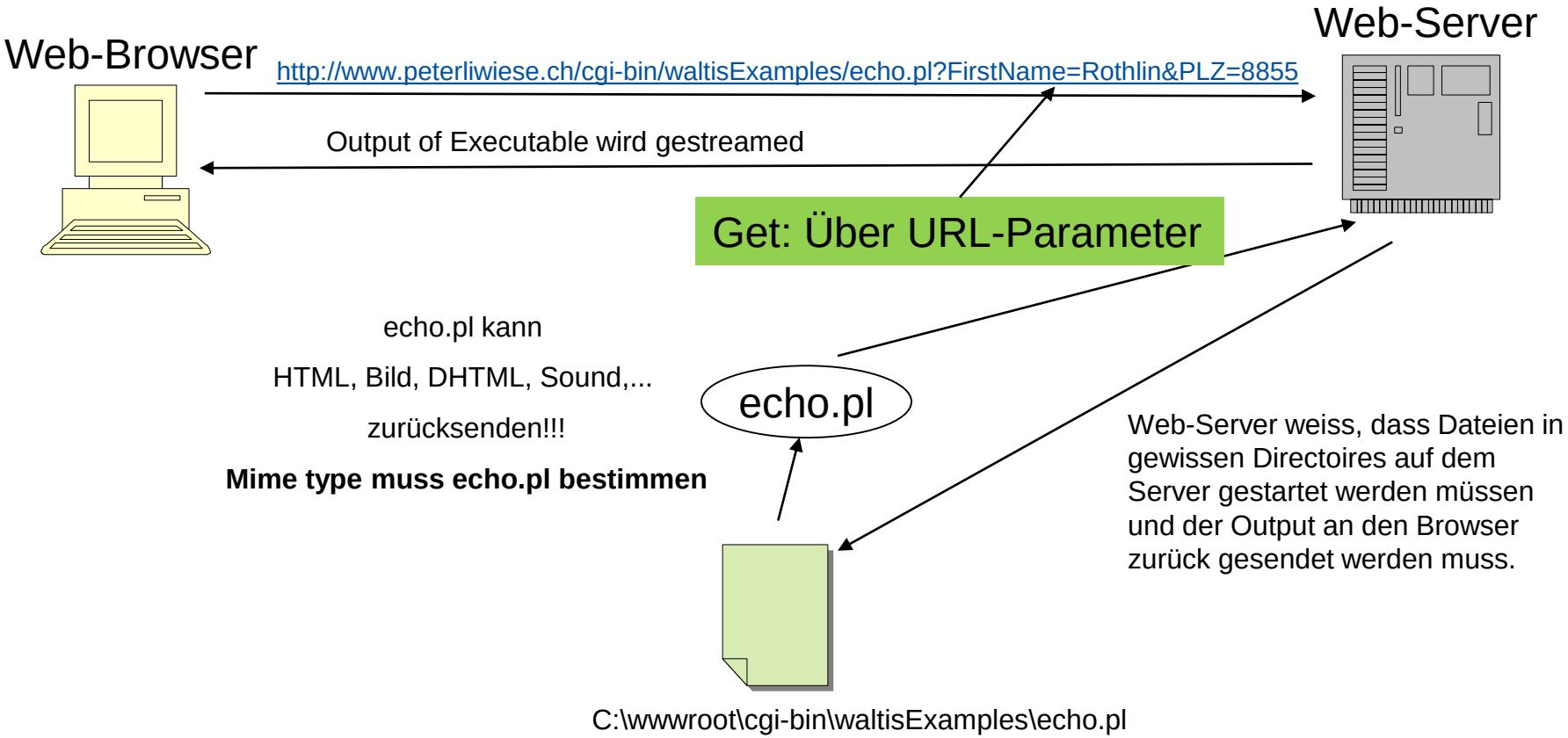
HTML Form-Elemente

http://www.peterliwiese.ch/AppletLib/formulare/formularElemente.html

TEXTAREA (with Wrap) Default Text	TEXTAREA (with Wrap) <TEXTAREA NAME=Textarea ROWS=3 COLS=50 WRAP=virtual>Default Text</TEXTAREA>
TEXTAREA (with Wrap) Default Text 1	TEXTAREA (with Wrap) <TEXTAREA NAME=Textarea1 ROWS=5 COLS=50 WRAP=Off>Default Text 1</TEXTAREA>
Reset Submit	<INPUT TYPE=RESET VALUE="Reset"> <INPUT TYPE=SUBMIT NAME=submitted VALUE="Submit">> </FORM>

Vollständige Definitionen: http://www.w3.org/TR/html401/interact/forms.html

Dynamic HTTP-Roundtrip



Nachteil:

Bei jedem Request wird neuer Prozess gestartet (Resourcen, Stateless)

URL-Parameter / HTML-Formulare

Unterschied POST und GET

http://www.peterliwiese.ch/AppletLib/formulare/echo.html

Name: Ein Text mit Ü ü ä

Text:

Absenden mit POST

```
<form action="/cgi-bin/waltisExamples/echo.pl" method=post>
  Name: <input size=40 maxlength=40 name="AnwenderName_1" Value="Ein Text mit &Uuml; &uuml; &auml;>
  Text: <textarea rows=5 cols=30 name="Kommentartext_1" wrap=virtual></textarea>
  <input type=submit value="Absenden mit POST">
</form>
```

Post: Über Header-Parameter

Name: Ein Text mit Ü ü ä

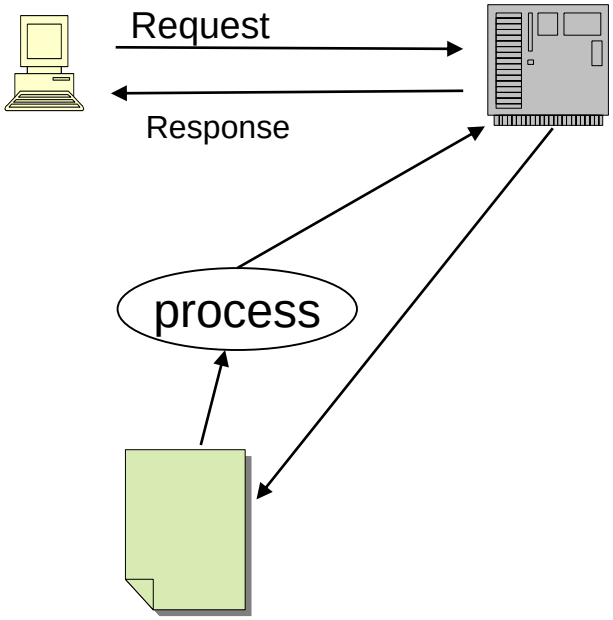
Text:

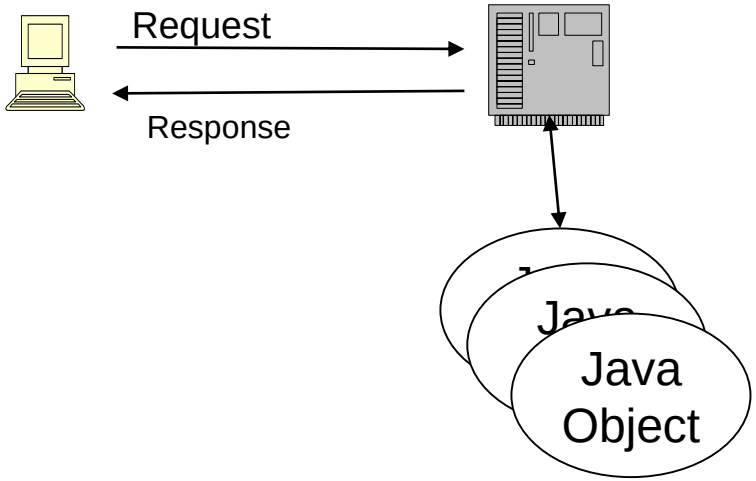
Absenden mit GET

```
<form action="/cgi-bin/waltisExamples/echo.pl" method=get>
  Name: <input size=40 maxlength=40 name="AnwenderName" Value="Ein Text mit &Uuml; &uuml; &auml;">
  Text: <textarea rows=5 cols=30 name="Kommentartext" wrap=virtual></textarea>
  <input type=submit value="Absenden mit GET">
</form>
```

Get: Über URL-Parameter

Servlet: Cgi versus Java way





- Parameter handling
- Return String
- Process creation
- Not object oriented
- No sessions and states

+ Objects already created
+ Nice interface
+ Proper Parameter handling in / out
+ Can handle states and sessions

Java-Script (einbinden)

```
// extern JS file
<script src="JS_Library.js" type="text/javascript"></script>

// code in Header
<script language="JavaScript">
<!-- Hide Script from older Browsers.
function saveValuesInCookies(form) {
    setCookie('Name',form.Name.value)
    setCookie('Vorname',form.Vorname.value)
    setCookie('Wohnort',form.Wohnort.value)
    setCookie('PLZ',form.PLZ.value)
    return;
}
//-->
</SCRIPT>
```

Java-Script (Event-Handlers)

Aus einem HTML-Link

```
<A HREF="javascript:saveValuesInCookies(document.adressForm);">Save Values </A>
```

Event-Handler auf Formelement

http://de.selfhtml.org/javascript/sprache/eventhandler.htm

```
<input type="button" onclick="checkBeforeSubmit(document.adrForm)" value="Submit">
<textarea rows="5" cols="40" onchange="alert(this.value)">Ein Text<textarea>
```

Event-Handler im BODY-TAG

```
<body onload="getValuesFromCookies(document.adressForm)">
```

Java-Script (code)

```
// function to check form content before sending to server
// -------------------------------------------------------
function checkBeforeSubmit (form) {
   if (!check(form)) return;
   form.submit();
   return;
}

function check(form) {
  form.Name.value = form.Name.value.toUpperCase();
  if (form.Name.value.length != 7 ) {
     alert ("Name has to be exactly 7 char long ("+form.Name.value+")");
     form.Name.focus();
     return (false);
  }

  if ((form.PLZ.value.length != 4 ) || (isNaN(parseInt(form.PLZ.value)))) {
     alert ("PLZ must be a 4 digit number");
     form.PLZ.focus();
     return (false);
  }

  saveValuesInCookies(form)

  return (true);
}
```

Simulate stateful sessions: Cookies

Textinformation, die vom Server (im Header oder via Client-Script) an den Client übertragen wird.

Beim nächsten http-Request werden alle Cookies dieser Seite mit dem Request mitgesendet.

➔ kann nun vom Server ausgewertet werden (z.B. Session-Id, welche dann zum Warenkorb führt)

"Set-Cookie:" Name "=" Wert *(";" Attribut)

Name=Wert ist eine Folge von encoded Zeichen. z.B. SessionId=12345678

Folgende Attribute sind in der Spezifikation von Netscape definiert:

- EXPIRES=*dateValue*
  Nach diesem Datum / Zeit wird Cookie gelöscht und nicht mehr gesendet
- DOMAIN=*domainName* (Default current Domain)
  Cookie wird nur mitgesendet bei Requests an diesen Domain-Name und deer Sub-Domains
- PATH=*pathName* (Default current Path)
  Cookie wird nur mitgesendet bei Requests an diesen Path
- SECURE
  Cookie wird nur gesendet, falls der Request über https geht

Simulate stateful sessions: Cookies (via JavaScript)

```
function setCookie(name, value) {
    var today = new Date();
    var expire = new Date();
    expire.setTime(today.getTime() + 1000*60*60*24*365);  // Cookie lebt 1 Jahr
    document.cookie = name + "=" + escape(value) + ";expires=" + expire.toGMTString();
}

function getCookie(name,defaultValue) {
    var search = name + "=";
    if (document.cookie.length > 0) {
        offset = document.cookie.indexOf(search);
        if (offset != -1) {
            offset += search.length
            end = document.cookie.indexOf(";", offset);
            if (end == -1)
                end = document.cookie.length;
            return  unescape(document.cookie.substring(offset, end));
        } else {
            return defaultValue;
        }
    } else {
        return defaultValue;
    }
}
```

Setzen:

setCookie('emailAdr', form.emailAdr.value)

Lesen:

form.emailAdr.value = getCookie('emailAdr','')

Simulate stateful sessions: Hidden Parameter

Dieses Beispiel: http://www.peterliwiese.ch/FormHandler/testFormBW.html

```
<BR>
e-mail Addresse
<INPUT TYPE=TEXT     NAME=emailAddr                VALUE="Walter@Rothlin.com">
<INPUT TYPE=HIDDEN   NAME=constrain_emailAddr      VALUE="isEmailAddr(XXXX,Keine g&#252;ltige e-mail Addresse)">
<BR>

<HR>
Kommentar (with Wrap)<BR>
<TEXTAREA NAME=Kommentar ROWS=10 COLS=70 WRAP=virtual>Default Text</TEXTAREA>
<BR>

<HR>
Kommentar 2 (no Wrap)<BR>
<TEXTAREA NAME=Kommentar_2 ROWS=5 COLS=50 WRAP=Off>Default Text</TEXTAREA>

<BR>
<HR>

<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_replyTemplateSuccess    VALUE="http://www.peterliwiese.ch/FormHandler/testSuccessTemplate.html">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_replyTemplateNotSuccess VALUE="http://www.peterliwiese.ch/FormHandler/testNotSuccessTemplate.html">



<!--
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_replyTemplateSuccess    VALUE="testSuccess.html">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_replyTemplateNotSuccess VALUE="testNotSuccess.html">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailTemplate           VALUE="test.txt">
-->
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailTemplate               VALUE="http://www.peterliwiese.ch/FormHandler/testEmailTemplate.html">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailSubject                VALUE="Antwort (Default as HTML)">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailToAddrs                VALUE="Walter@Rothlin.com">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailCcAddrs                VALUE="">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailBccAddrs               VALUE="">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailFromAddr               VALUE="FormField:emailAddr">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailSentAsHTML             VALUE="YES">

<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailTemplate_1             VALUE="http://www.peterliwiese.ch/FormHandler/testEmailTemplate.txt">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailSubject_1              VALUE="Antwort as Normal">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailFromAddr_1             VALUE="Walter@Rothlin.com">
<INPUT TYPE=HIDDEN NAME=cgiCmdHidden_emailSentAsHTML_1           VALUE="NO">
```

Beispiel «Ausgefüllter Einzahlungsschein»

Test-Bestellformular Tickets

Anzahl Tickets à CHF 45.-: 2

Lieferung:
- (●) Per eingeschriebener Post (+ CHF 6.00)
- ( ) Abholen bei Daniel Vogt, 8855 Wangen

Rechnungsbetrag (CHF): 96.00

Ihre Angaben

Name: ROTHLIN

Vorname: Walter

Strasse Nr: Peterliwiese 33

PLZ: 8855 Ort: Wangen

email: walter@rothlin.com

Kommentar

Test (sollte nicht bei Simon ankommen, sonst bitte sofort melden)

Kostenpflichtig bestellen

Rechnung Ticket-Bestellung Oktoberfest Wangen SZ

Rechnungs-Nummer:**001029** (Bitte bei Zahlung angeben)

2 Ticket(s) zu CHF 45.--

Lieferung: **Wird nach Eingang der Zahlung per Eingeschriebener-Post gesendet plus CHF 6.00**

Zu bezahlender Betrag:------**96.00 CHF**

Ihre Kommentar:
Test (sollte nicht bei Simon ankommen, sonst bitte sofort melden)

Empfangsschein / Récépissé / Ricevuta

Einzahlung für / Versement pour / Versamento per
Schwyzer Kantonalbank
6431 Schwyz

Zugunsten von / En faveur de / A favore di
CH41 0077 7005 8249 1145 5
Feuerwehrverein Wangen-Nuolen
8855 Wangen SZ

Konto / Compte / Conto 60-1-5
CHF – – – – – – 9 6 . 0 0

Einbezahlt von / Versé par / Versato da
Walter ROTHLIN
Peterliwiese 33
8855 Wangen

Die Annahmestelle
L'office de dépôt
L'ufficio d'accettazione

Einzahlung Giro / **Versement Virement** / **Versamento Girata**

Einzahlung für / Versement pour / Versamento per
Schwyzer Kantonalbank
6431 Schwyz

Zugunsten von / En faveur de / A favore di
CH41 0077 7005 8249 1145 5
Feuerwehrverein Wangen-Nuolen
8855 Wangen SZ

Konto / Compte / Conto 60-1-5
CHF – – – – – – 9 6 . 0 0

303

Zahlungszweck / Motif versement / Motivo versamento
Oktoberfest: 001029

Einbezahlt von / Versé par / Versato da
Walter ROTHLIN
Peterliwiese 33
8855 Wangen

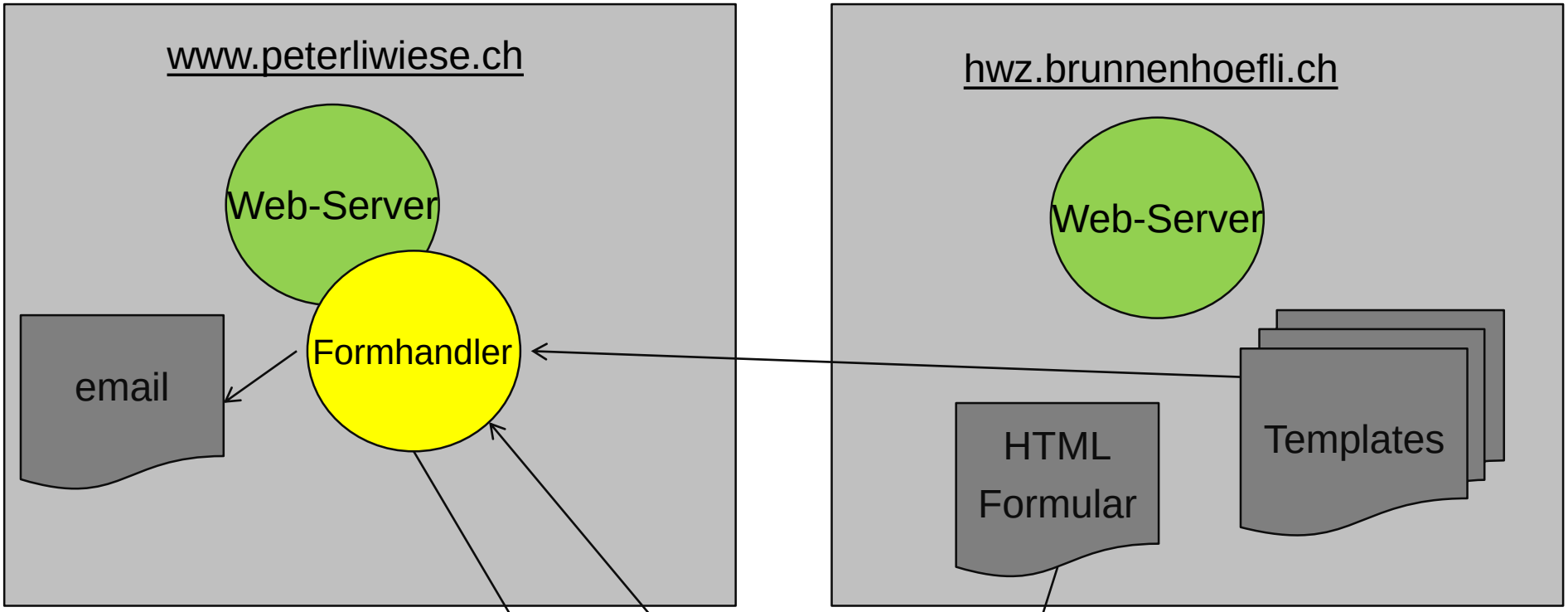
0000000000000000582491145 59+ 070077709>
600000015>

Vielen Dank für Ihre Bestellung

http://www.peterliwiese.ch/oktoberfest/TestBestellformular.shtml (Beispiel zum Ausprobieren)

http://www.hwz.brunnenhoefli.ch/uebungen.html (Beispiel 1.6 für Source-Code)

Übung Formularhandler



Client-Browser:

HWZ Hochschule für Wirtschaft Zürich

Java Servlets: Struktur

```
public class SimpleHttpServlet extends HttpServlet {
    protected void doGet( HttpServletRequest req, HttpServletResponse res)
                 throws ServletException, IOException {
                  doPost(req, res);
    }
    protected void doPost( HttpServletRequest req,HttpServletResponse res)
                 throws ServletException, IOException {
                 response.setHeader("Content-Type", "text/plain");
                 response.getWriter().write("GET/POST response");
    }
    public void init(ServletConfig servletConfig) throws ServletException {
         System.out.println("init() is called!");
    }
    public void destroy() {
         System.out.println("destroy() is called!");
    }
}
```

Java Servlets: Zugriff auf Parameter und Response

```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response)
                 throws ServletException, IOException {

        // Parameter lesen
        String param1 = request.getParameter("param1");
        String contentLength = request.getHeader("Content-Length");

        // Antwort schreiben
        response.setHeader("Content-Type", "text/plain");
        // oder response.setContentType("text/html");
        PrintWriter writer = response.getWriter();
        writer.print("Param1: " + param1);
}
```

Doc-Pages:

http://docs.oracle.com/javaee/7/api/javax/servlet/ServletRequest.html

http://docs.oracle.com/javaee/7/api/javax/servlet/http/**HttpServletRequest**.html

http://docs.oracle.com/javaee/7/api/javax/servlet/ServletResponse.html

http://docs.oracle.com/javaee/7/api/javax/servlet/http/**HttpServletResponse**.html

Java Servlets: HttpSession

```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response)
               throws ServletException, IOException {

       HttpSession session = request.getSession();

       //setzen eines Wertes in der Session
       session.setAttribute("userName", "theUserName");

       //lesen eines Wertes aus der Session
       String userName = (String) session.getAttribute("userName");
}
```

Java Servlets: Cookies

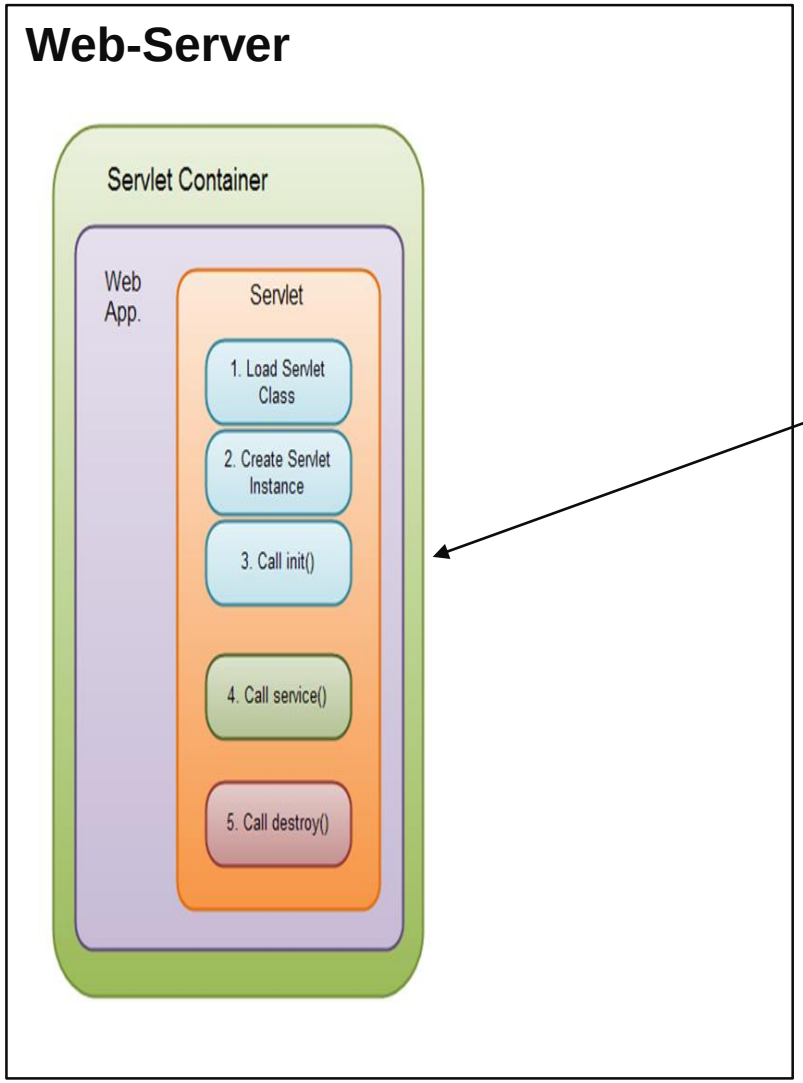
```
:
 // Setzen eines Cookies
 Cookie aCookie = new Cookie("myCookie", "myCookieValue");

 aCookie.setValue("myCookieValue");
 aCookie.setComment("Ein Kommentar");


 // cookie wird mitgesendet, an alle requests auf "rothlin.com" und deren sub-domains WICHTIG .

 aCookie.setDomain(" . rothlin.com");


 // Cookie wird mitgesendet auf alle requests "unter" /stat
 aCookie.setPath("/stat");

 aCookie.setMaxAge(seconds);  // if seconds < 0 then Deleted after browser is closed
                              // if seconds = 0 then Cookie is deleted
 response.addCookie(aCookie);


 // Lesen aller Cookies
 Cookie[] cookies = request.getCookies();


:
```

Max. 4kB data

http://docs.oracle.com/javaee/6/api/javax/servlet/http/Cookie.html

Nicht alltägliches Beispiel, was mit Cookies machen kann: http://www.homepage-performance.de/cookies-path.html

Java Servlets: Struktur



Servlet-Container ist Teil eines Applikation-Servers (z.B. Glassfish, Tomcat)

JSP: Java Server Page (First Example)

```
<%@ page import="java.util.*" %>
<html>
  <body>
    <%    // This is a scriptlet (java code! Therefore comments java-sytle.)
          System.out.println( "Evaluating date now" ); // goes to server log
          Date date = new Date();                      // java.util nicht nötig, da page import
          out.println("1: " + String.valueOf( date )); // goes to the HTML stream
    %>
    <BR/>Hello!  The time is now <%= date %> or <BR/>   <!--This is an expression -->
    <%
          out.println( date );
          out.println( "<BR>Your machine's address is " );
          out.println( request.getRemoteHost());
    %>
  </body>
</html>
```

HWZ_JSP_Bean:index.jsp

JSP-tutorial: http://www.jsptut.com/

Beans / Enterprise Java Beans (EJB)

- Bean/EJB ist ein POJO mit Business-Funktionen
- Bean befindet sich im gleichen Container wie Aufrufer
- EJB befindet sich in einem fremden Container
- EJB sind standardisierte Java Komponenten, welche innerhalb eines Containers laufen.
- Standardaufgaben werden vom Container übernommen und müssen nicht vom Entwickler programmiert werden
- Die Komponenten können Remote (von ausserhalb des Containers) oder Local (von einer anderen Bean innerhalb des gleichen Containers) angesprochen werden

Es gibt folgende Kategorien von EJB:

- **Session Beans**
  - Stateless (Kontainer kann diese mehrmals instanzieren und wiederverwenden)
  - Stateful (Keeps states within calls during the same session)
  - Singleton (Kontainer kreiert nur ein Objekt davon und verwendet immer das Gleiche)
- **Massage Driven Beans (MDB)**
- Entity Beans (seit J2EE 5 / EJB 3.0 keines mehr sondern ein POJO mit Annotations [Plain Old Java Object])
  - Persistenz via Datentransfer-Objects (DTO)
  - Wird in einer RDB abgespeichert (Ein Object representiert eine row in table)
  - OR-Mapping like Hibernate

Bean oder Enterprise-Java Bean (EJB)

```
import javax.ejb.Stateless;
@Stateless
public class HelloWorldBean {

   @PostConstruct
   public void initialize() {
      System.out.println("PostConstruct von HelloBean aufgerufen!");
   }
   @PreDestroy
   public void cleanup() {
      System.out.println("PreDestroy von HelloBean aufgerufen!");
   }

   public String getHello() {
         return "Hello World !";
   }
}
```

Bean oder EJB
Bei der Erstellung kein Unterschied

HelloWorldBean.java

HWZ Hochschule für Wirtschaft Zürich

Servlet ruft Bean (im gleichen container) auf

```
import java.io.*; import javax.ejb.EJB;
import javax.servlet.*; import javax.servlet.htteServlet;

public class TestServlet extends HttpServlet {
      @EJB
      private HelloWorldBean helloWorld;

      public void service (HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException {
              resp.getWriter().println(helloWorld.getHello());
      }
}
```

Servlet welches das Bean verwendet

TestServlet.java

Servlet ruft Enterprise-Java Bean EJB (in anderem container) auf

```
CalculatorBeanRemote calcBean = null;

public void init(ServletConfig servletConfig) throws ServletException {
    System.out.println("SimpleServlet init() is called!");
    System.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
     try {
       InitialContext ctx = new InitialContext();
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", "3700");
       calcBean = (CalculatorBeanRemote) ctx.lookup(
         "java:global/HWZ_SessionBean/CalculatorBean!com.rothlin.hwz.CalculatorBeanRemote");
     }
     catch (Exception ex) {
        System.out.println("Couldn't create converter bean!" + ex);
     }
}
```

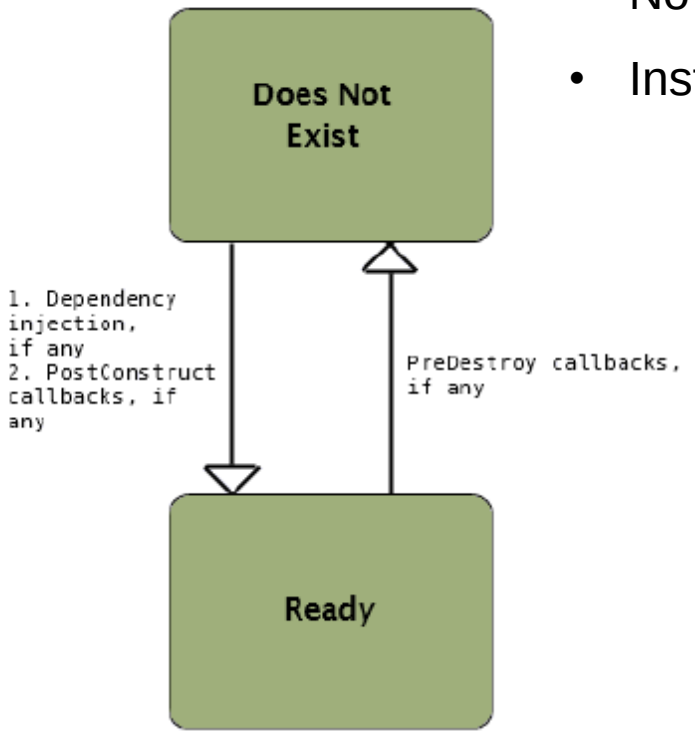
Servlet holt Objekt-Referenz vom EJB über einen Nameservice (JNDI)

Beim Deployment vom EJB wird automatisch ein JNDI eintrag gemacht und der Name wird in der Console angezeigt

HWZ Hochschule für Wirtschaft Zürich

EJB: Stateless Session Beans

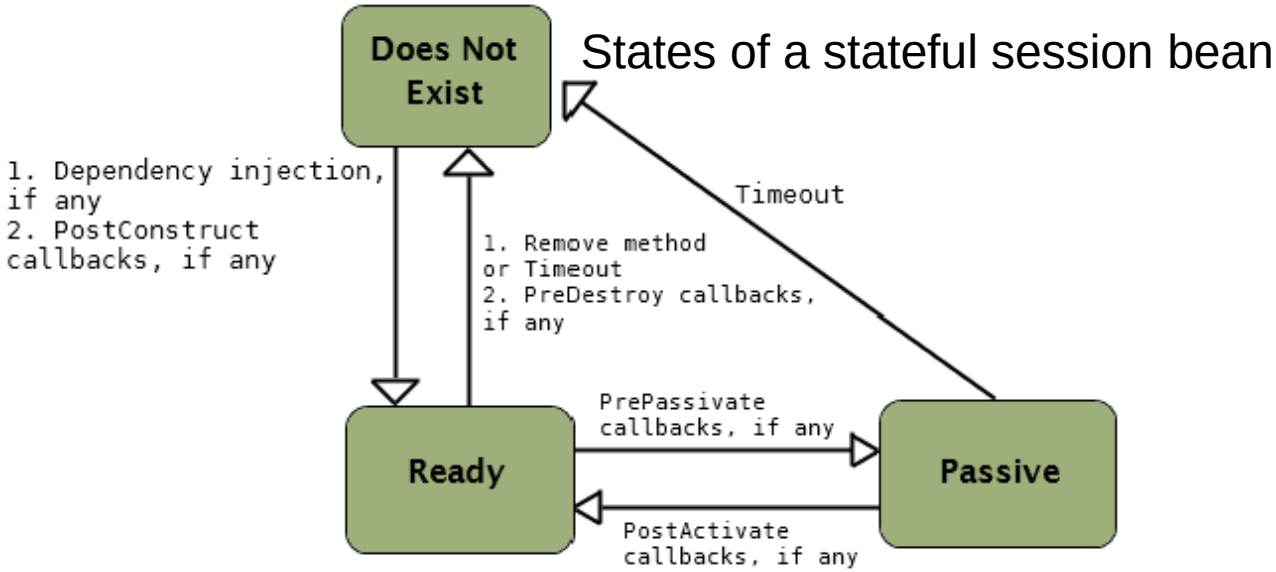
- Performs an operation for the client
- No associated conversational state
- May have instance state
- No concurrent access
- Instance variables do not keep the values across calls



States of a stateless session bean

EJB: Stateful Session Beans

- Performs an operation for the client
- Associated conversational state (**per sessions**)
- Instance variables represent the state
- Instance variables keep the values across calls (within the same session)
- **Container-managed** persistence(CMP) beans: The logic for synchronizing the bean's state with the database is handled automatically by the container. B**ean-managed** persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container.



Client to call EJB

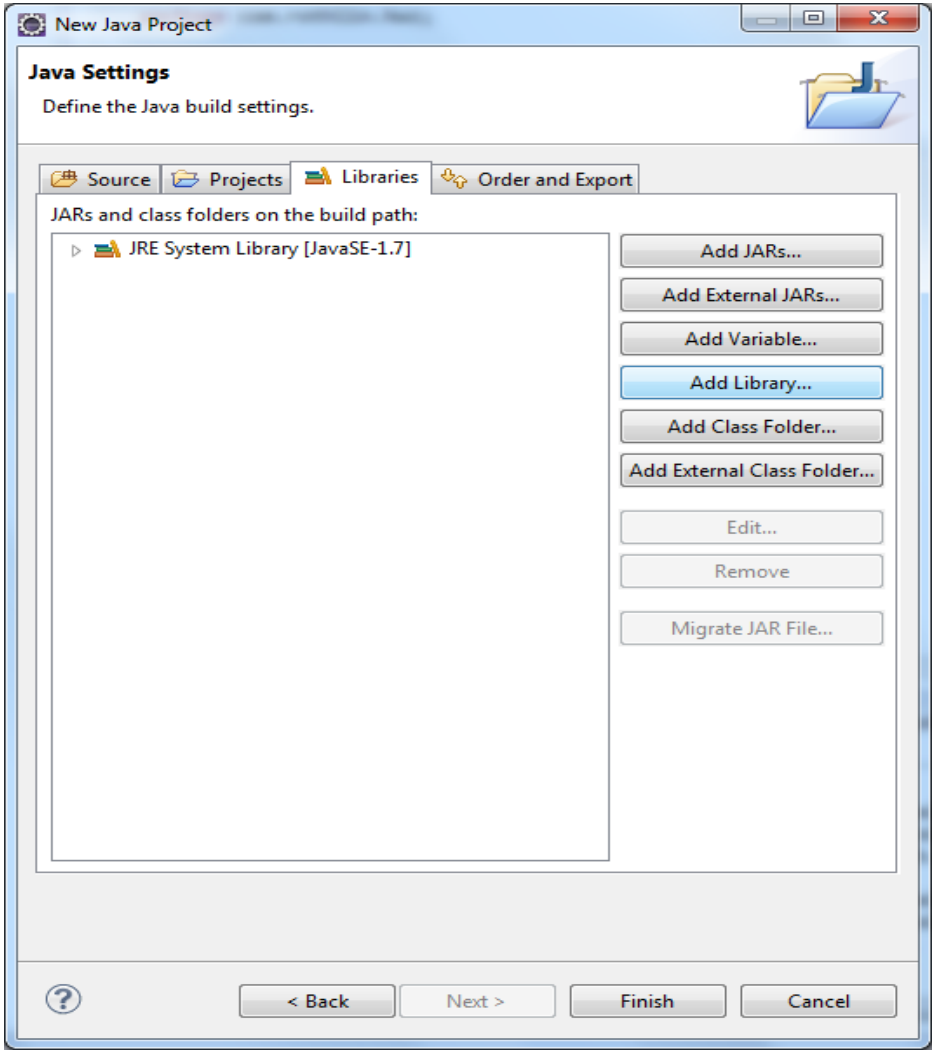
```
import javax.naming.InitialContext;
public class BeanClientMain {
   public void runTest() throws Exception {
       System.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
       InitialContext ctx = new InitialContext();

       System.out.println("Calling Calc Bean (Session stateless)");
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", "3700");
       CalculatorBeanRemote bean = (CalculatorBeanRemote) ctx.lookup
              ("java:global/HWZ_SessionBean/CalculatorBean!com.rothlin.hwz.CalculatorBeanRemote");

        int result = bean.multInt(4, 5);
        System.out.println("4 * 5 = " + result);
   }

   public static void main(String[] args) {
         BeanClientMain cli = new BeanClientMain();
         try { cli.runTest(); }
         catch (Exception e) { e.printStackTrace(); }
   }
}
```

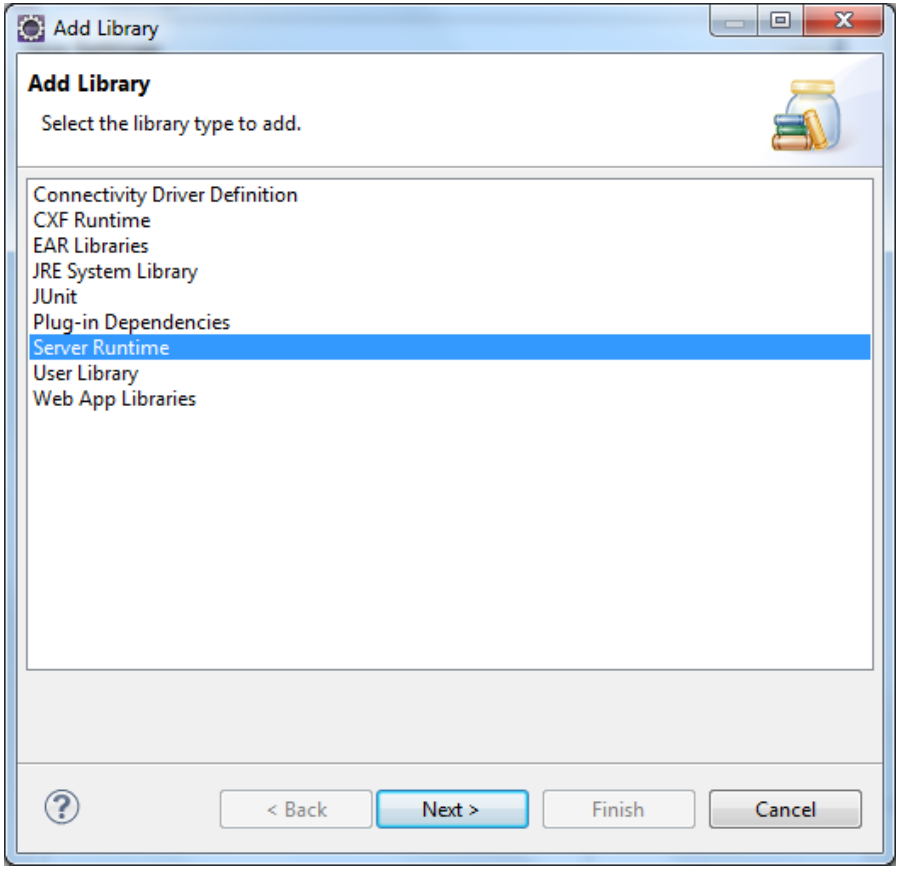
Nach dem Deployment des Beans auf den Application-Server findet man diesen String im Log-File

BeanClientMain.java

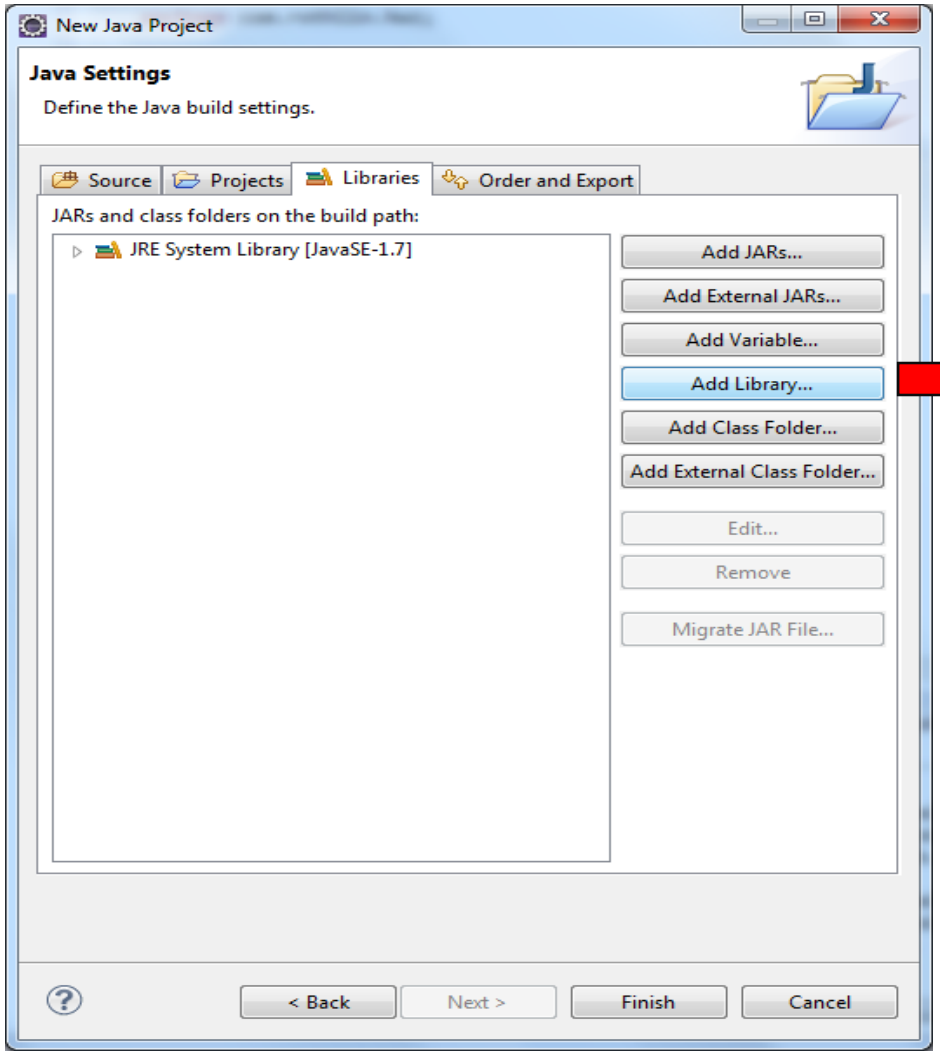
Übung: Stateless Bean

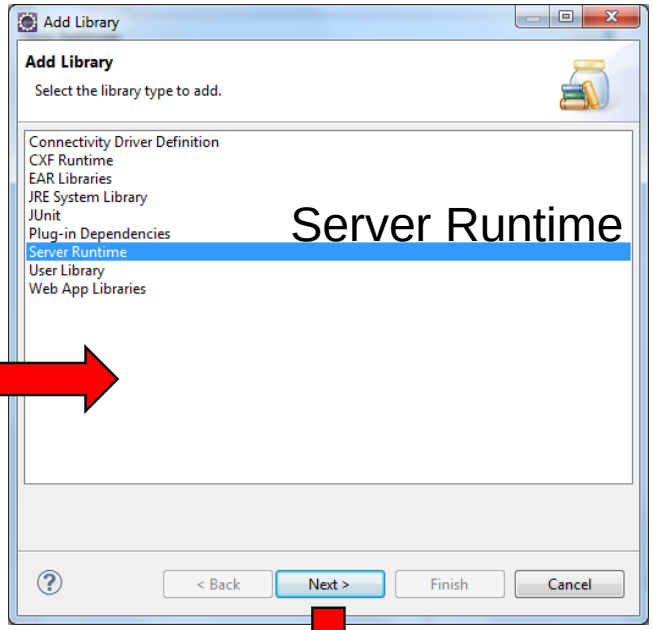
Add Libraries (Server Runtime)

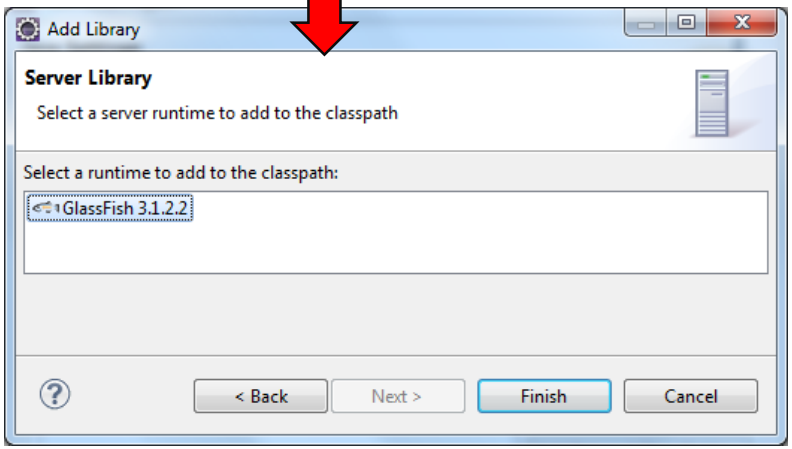




Übung: Stateless Bean

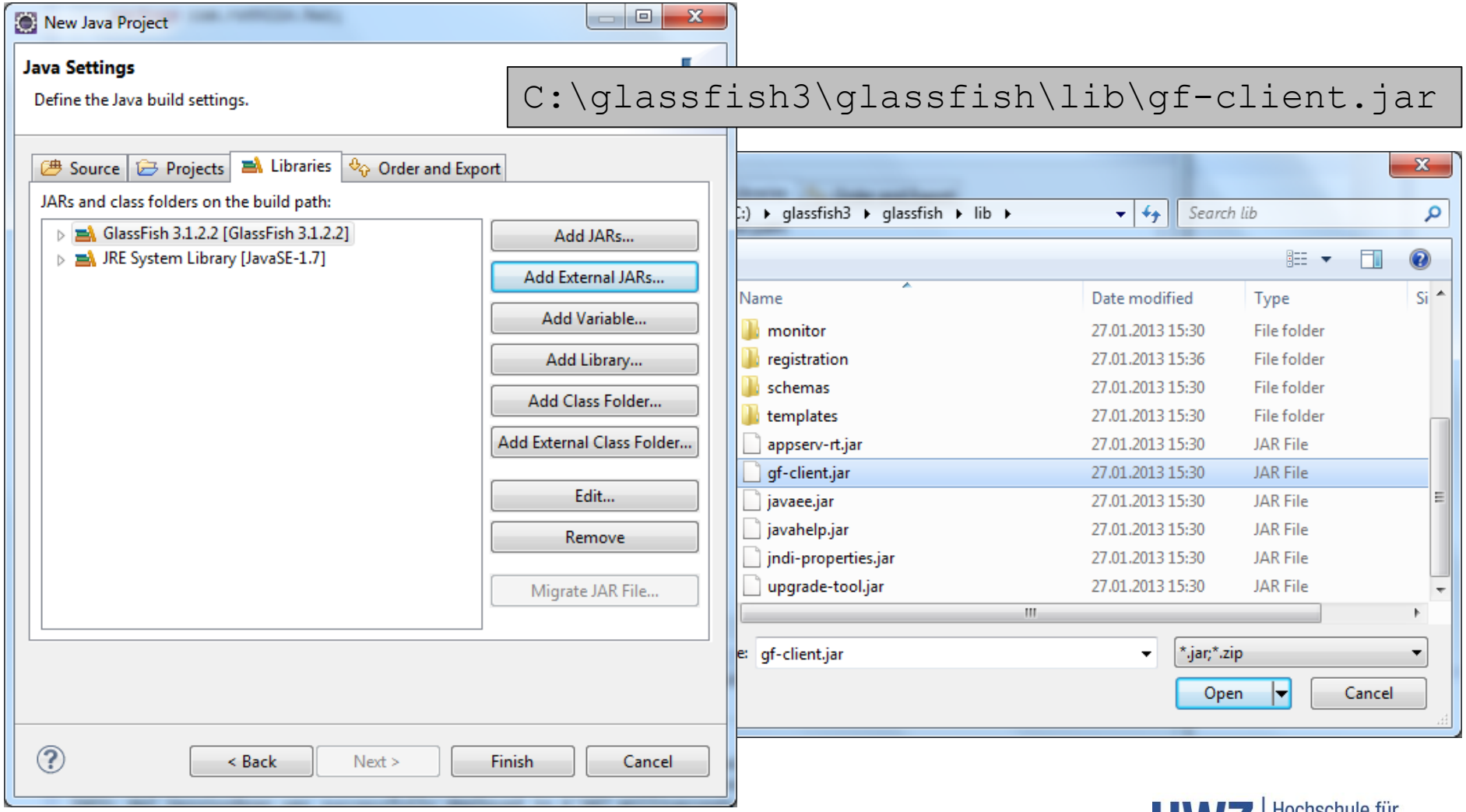




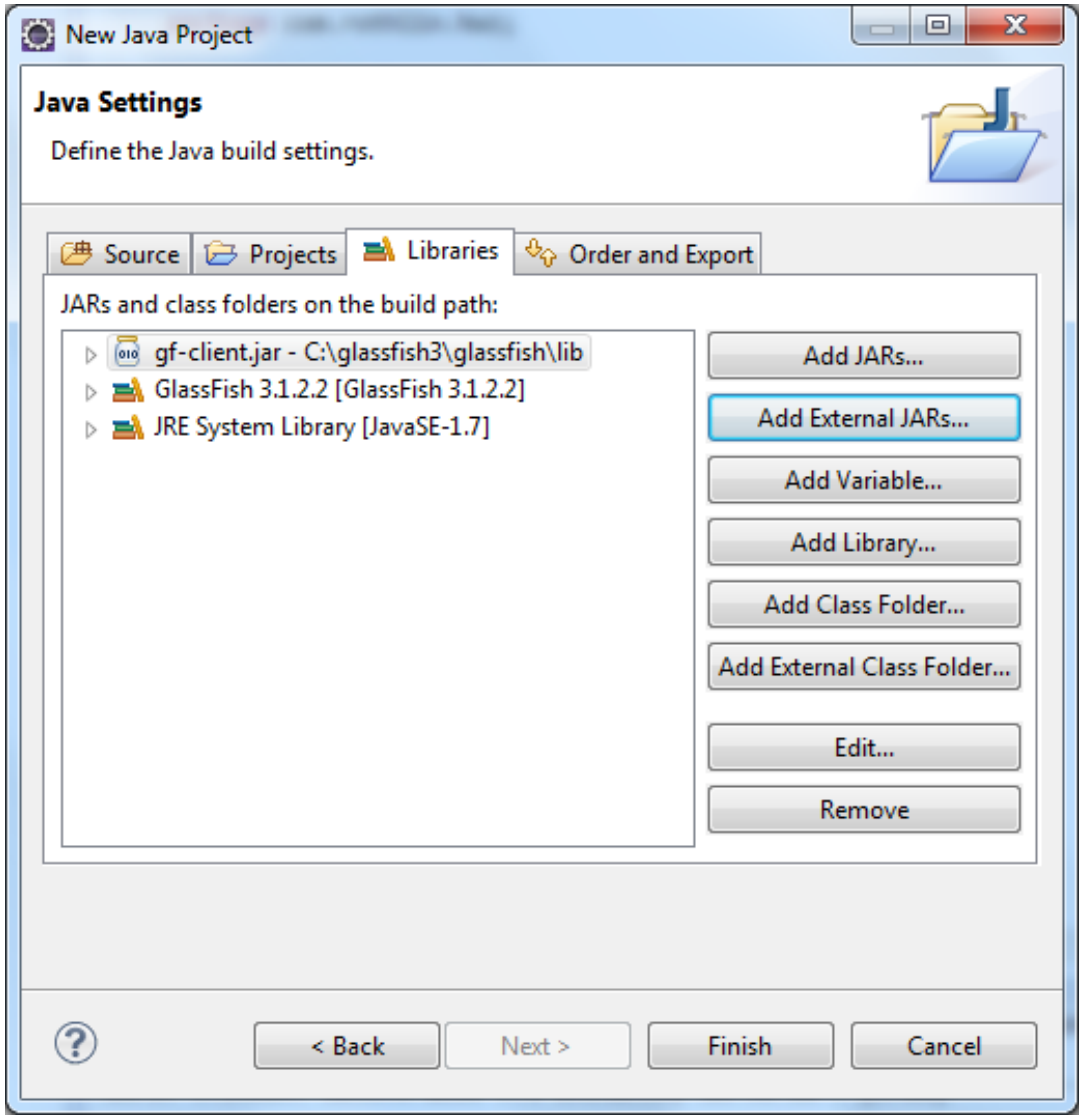


Übung: Stateless Bean

Add External JAR



Übung: Stateless Bean



DB-Connections von JAVA

Connection String: <Prefix>:<DB Type>://<Server>:<Port>/<Database>

Beispiel: jdbc:mysql://localhost:3306/Example

Connection String	
Prefix	jdbc
DB Type	Mysql
Server	localhost
Port	3306
Database	Example

Connection Properties	
Username	root
Password	root

Verbindung herstellen

Connection dbc=DriverManager.getConnection(connectionString, connectionProperties);

Queries ausführen

Statement stmt = dbc. createStatement();

ResultSet rs = stmt.executeQuery(queryString);

Das Ergebnis der Query ist anschliessend in einem ResultSet. Das ResultSet besteht – wie der Name vermuten lässt - aus verschiedenen Resultaten. Eine Row ist ein Resultat und kann auf Column Ebene angesprochen werden.

HWZ Hochschule für Wirtschaft Zürich

DB-Connections von POJO

```
public class DB_Connection_JDBC {
  static Connection dbConnection;

  public static void main(String[] args) {

     try { Class.forName("com.mysql.jdbc.Driver");   // this is not always neccessary; but doesn't harm
     } catch (ClassNotFoundException e1) { e1.printStackTrace(); }

     Properties connectionProperties = new Properties();
     connectionProperties.put("user",     "studenten");
     connectionProperties.put("password", "hwz007");

     try { dbConnection = DriverManager.getConnection("jdbc:mysql://localhost:3306onlineshop",connectionProperties);
            Statement stmt = dbConnection.createStatement();
            ResultSet rs = stmt.executeQuery("Select * from kunden;");  // SQL-Query
            int id; String name; int age;

            System.out.println("ID|Name|Age");
            while (rs.next()) {
               id   = rs.getInt("id");
               name = rs.getString("name");
               age  = rs.getInt("age");
               System.out.println(id + "|" + name + "|" + age);
            }
     } catch (SQLException e) { e.printStackTrace();}
  } // end main
} // end class
```

DB-Connections von Servlet über JNDI

```
private static Connection dbConnection;

public void init(ServletConfig servletConfig) throws ServletException {
    Context ctx;
    try {
           ctx = new InitialContext();
           DataSource ds = (DataSource) ctx.lookup("jdbc/onlineshop");
           dbConnection = ds.getConnection();
    }
    catch (NamingException e2) { e2.printStackTrace(); }
    catch (SQLException e)     {  e.printStackTrace(); }

    if (dbConnection  != null) { System.out.println("Connection established via JNDI<BR/>"); }
                          else { System.out.println("Connection failed via JNDI<BR/>");      }
}

public void destroy() {
    try {
           dbConnection.close();
    }
    catch (SQLException e) { e.printStackTrace(); }
}
```

DB-Connections im JNDI einrichten

Path 2 (Scripts zur Vertiefung)

Server-Side scripting (CGI Common Gateway Interface)

Server Side: Exec and SSI tags

In einer HTML Seite werden an bestimmten Stellen Server Side Code aufgerufen. Der Server ersetzt in der Orginalseite den Aufruf mit dem Return-Value des Aufrufes.

Der Server muss wissen, in welchen Datei nach EXEC / SSI Tags geparst werden muss *.shtml

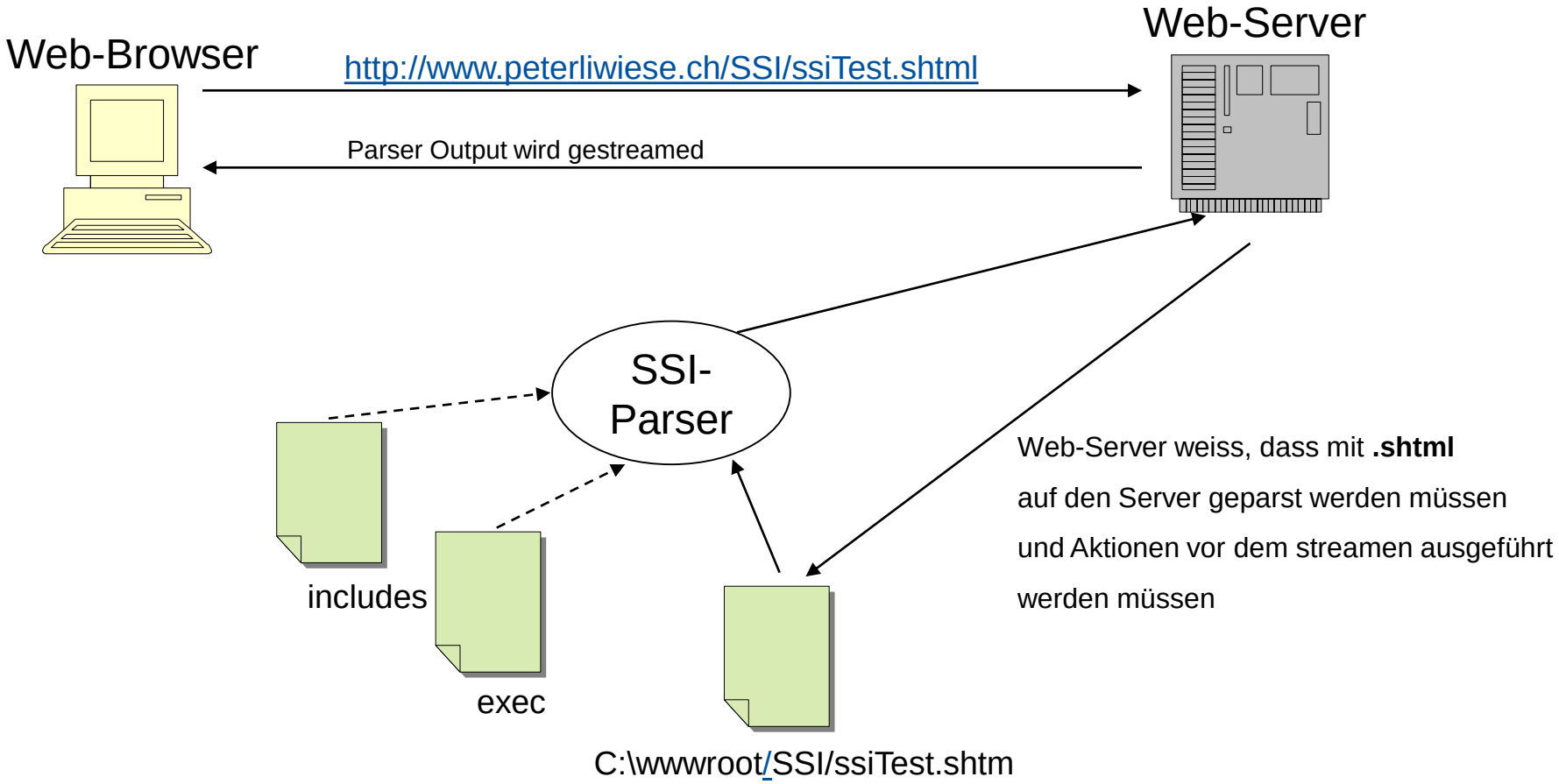
Beispiel:

http://www.peterliwiese.ch/SSI/ssiTest.shtml

http://www.peterliwiese.ch/Counter/index.shtml

```
Ein Beispiel eines Zaehlers<BR><BR><BR>
<!--#include virtual="http://www.peterliwiese.ch/cgi-bin/Counter/siZaehler.pl?counterId=test"-->
```

Server Side Includes (SSI): Roundtrip



Server Side: Weitere Programming Möglichkeiten

Möglichkeiten für Server Side Programming:

- cgi (common gateway interface)
- Servlet und JSP (Java server page)
- ASP (Active server page)
- Ajax: http://www.peterliwiese.ch/Ajax/first.html
- php (personal home page) http://www.peterliwiese.ch/PHP_Examples/time.php3

- Sprache muss vom Server verstanden werden
- Kein Code beim Client (Kopierschutz)
- Kann auf DB oder andere Daten auf dem Server zugreifen

Empfänger auf Server-Seite

```perl
sub echo {
  print("<html><head><title>CGI-Feedback</title></head>\n");
  print("<body><h1>CGI-Feedback</h1>\n");
  if ($ENV{"REQUEST_METHOD"} eq "GET") {
    print("Daten wurden via <B>QUERY_STRING (GET)</b> uebertragen<br>\n");
    printf("<STRONG>Query-String:</STRONG> %s<br>\n",    $ENV{"QUERY_STRING"});
  } else {
    print("Daten wurden via <B>Standart-Input (POST)</b> uebertragen<br>\n");
  }
  if (isItProbablyAnMS_Browser()) {
    print("You probably use an Microsoft Browser\n");
  } else {
    print("You use another Browser than a Microsoft\n");
  }
  print("<HR><H3>Request spezifische Informationen</H3>");
  displayHashTableHTML(%ENV);
  print("<HR><H3>Parameters passed</H3>\n");
  my(@names) = getParameterNames();
  my($aName) = "";
  my(%paramsPairs) = ();
  foreach $aName (@names) {
    my($aValue) = param($aName);
    %paramsPairs = (%paramsPairs,($aName,$aValue));
  }
  displayHashTableHTML(%paramsPairs);
  print("<HR>\n");
  print("</body></html>\n");
}
```

siPerlLib.pm

```perl
sub getParameterNames {
  my($pattern) = @_;
  my($query) = new CGI;
  my(@names) = $query->param;
  if ($pattern ne "") {
    my @tmpNames = ();
    foreach $aParamName (@names) {
      if (index($aParamName,$pattern) != -1) {
        push(@tmpNames,$aParamName);
      }
    }
    @names = @tmpNames;
  }
  return @names;
}

sub isItProbablyAnMS_Browser {
   my($userAgent) = $ENV{"HTTP_USER_AGENT"};
   return ($userAgent =~ /MSIE/);
   HTTP_USER_AGENT
}
```

siPerlLib.pm

```perl
#!/usr/bin/perl

use CGI qw(param);
use siPerlLib;

print "Content-type: text/html\n\n";

echo();
```

echo.pl

HWZ Hochschule für Wirtschaft Zürich

Kombination von Server Side und Client Side Programming

Ein Server Programm erstellt eine dynamische HTML Seite (abhängig von Daten in einer DB) mit JAVA-Script, welche die Seite dynamisch auf der Client-Seite verändern lässt!

http://www.naturfreunde-lachen.ch ➔ Adress-Verwaltung

UserId: Walter_Rothlin Password:

Übung Server-Side Scripting

Aufgabe 2.1 – 2.4: http://www.hwz.brunnenhoefli.ch/

- Erstellen Sie ein Server-Script (ksh), welches verschiedenen Details vom Server in einer HTML-Page darstellt.
- Ändern Sie Ihre Klassen-Adressliste so, dass die Adressen von einem Server-Script aus einem csv-File gelesen und angezeigt werden (Perl oder ksh).
- Erstellen Sie eine einfache Gästebuch Web-Applikation (Formular + Server-Script)
- Erstellen Sie einen «http Web-Service» der die Adressen in einem csv File liefert. Greifen Sie mit einem VBA Script aus Excel heraus, auf diesen Service zu und laden die neusten Adressen.

Java Servlet

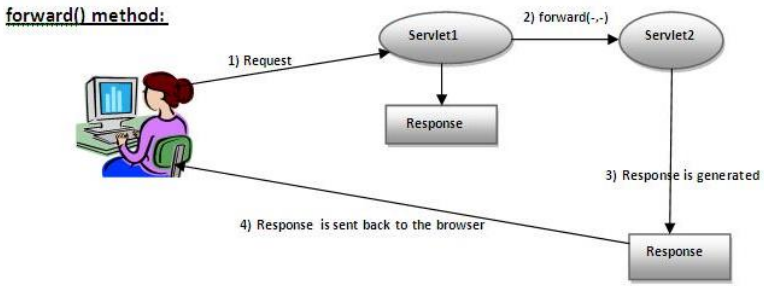
Java Servlets: response.Redirect()

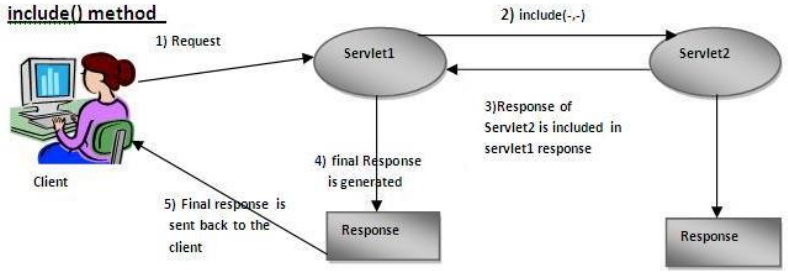
```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response)
                 throws ServletException, IOException {
        response.setHeader("Content-Type", "text/plain");
        PrintWriter writer = response.getWriter();
        writer.write("This is just plain text");
        System.out.println("Hallo 1");
        response.sendRedirect("http://hwz.brunnenhoefli.ch");
        writer = response.getWriter();
        writer.write("This is just plain text");
        System.out.println("Hallo 2");
}
```

- URL in response changes to redirected URL
- All writes before and after the redirect **are not in response**
- All statements before and after the redirect **are executed** (e.g. System.out.println)

Java Servlets: requestDispatcher.forward() requestDispatcher.include()

```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response)
                 throws ServletException, IOException {
        RequestDispatcher requestDispatcher =
                        request.getRequestDispatcher("/anotherURL.simple");
        requestDispatcher.forward(request, response); //no merging of response output
        requestDispatcher.include(request, response); //merging of response output
}
```

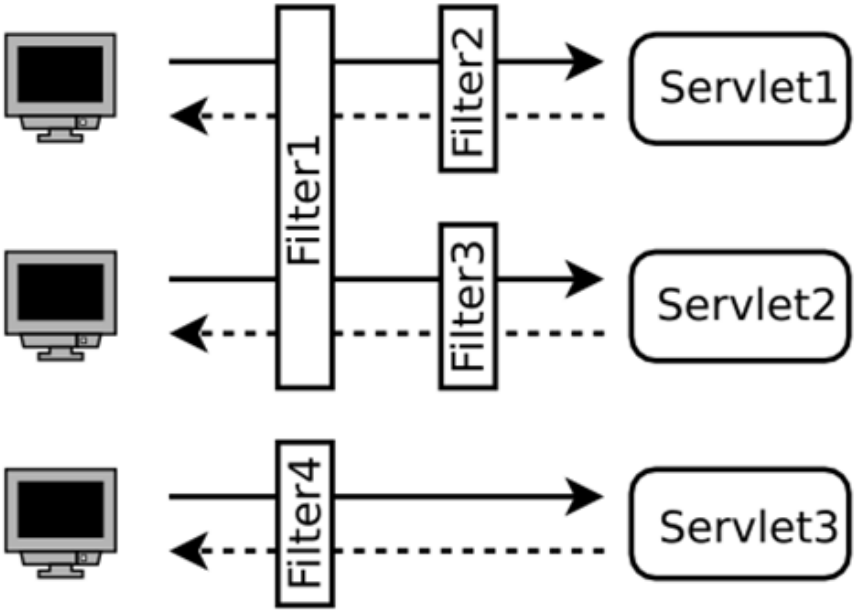




- URL not changed, stays to calling servlet
- forward: One response just from the forwarded URL
- Include: All URL content are part of the respones + writes from the servlet as well

HWZ Hochschule für Wirtschaft Zürich

Filters



Beispiele von Filtern:

- Loggen
- Verschlüsselung / Entschlüsselung
- Compress
- Transformation (z.B. XSLT)

➔ Separation of concerns

➔ Filter can be added via configuration (web.xml)

Filters (Beispiel)

http://viralpatel.net/blogs/tutorial-java-servlet-filter-example-using-eclipse-apache-tomcat/

```
package com.rothlin.hwz;
:
@WebFilter(servletNames = { "HelloWorldServlet" })
public class HelloWorldServletFilter implements Filter {
    public HelloWorldServletFilter() {}
    public void destroy() {}
    public void init(FilterConfig fConfig) throws ServletException {}
    public void doFilter(ServletRequest servReq, ServletResponse res, FilterChain chain)
                                throws IOException, ServletException {
          HttpServletRequest request = (HttpServletRequest) servletRequest;
          String ipAddress = request.getRemoteAddr();
          System.out.println("IP:" + ipAddress + ", Time: " + new Date().toString());
          chain.doFilter(servReq, res);
    }
}
```

HWZ_ServletFilter:HelloWorldServletFilter

Filters Configuration (Beispiel web.xml)

```
 <filter>
    <filter-name>LogFilter</filter-name>
    <filter-class>com.rothlin.hwz.HelloWorldServletFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>LogFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

HWZ_ServletFilter:web.xml

Java Servlets: HttpSession

```
protected void doGet( HttpServletRequest request,
                      HttpServletResponse response)
               throws ServletException, IOException {

       HttpSession session = request.getSession();

       //setzen eines Wertes in der Session
       session.setAttribute("userName", "theUserName");

       //lesen eines Wertes aus der Session
       String userName = (String) session.getAttribute("userName");
}
```

Java Servlets: Servlet Context

```
protected void doGet( HttpServletRequest request,HttpServletResponse response)
                throws ServletException, IOException {
        ServletContext context = request.getSession().getServletContext();

        //setzen eines Wertes im servlet context
        context.setAttribute("someValue", "aValue");

        //lesen eines Wertes aus dem servlet context
        Object attribute = context.getAttribute("someValue");
}
```

Servlet Context Attributes sind für alle Besucher der Web-Applikation zugreifbar.

Session Attributes nur für einen einzigen Besucher.

Java Servlets: Configuration web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>

    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.rothlin.SimpleServlet</servlet-class>
  </servlet>


  <servlet-mapping>
    <servlet-name>controlServlet</servlet-name>
    <url-pattern>xxx*.html</url-pattern>  <!-- * as wild card -->

  </servlet-mapping>

</web-app>
```

Maps the Java-Class to a URL by having a common <servlet-name>

http://tutorials.jenkov.com/java-servlets/web-xml.html

Java Servlets: (init and context parameter)

```
:
:
  <servlet>
:
    <init-param>
       <param-name>myInitParam</param-name> <param-value>InitValue_1</param-value>
    </init-param>


    <context-param>
       <param-name>myContextParam</param-name> <param-value>ContextValue_1</param-value>
    </context-param>
  </servlet>
:
```

Only for this component

For all components within this web-app

Only ONE param-name pair per tag

web.xml

```
public class SimpleServlet extends GenericServlet {
  protected String myInitParam = null;
  public void init(ServletConfig servletConfig) throws ServletException{
    this.myInitParam = servletConfig.getInitParameter("myInitParam");
  }
  public void service(ServletRequest request, ServletResponse response)
         throws ServletException, IOException {

    String myContextParam = request.getSession().getServletContext().getInitParameter("myContextParam");
    response.getWriter().write("<html><body> myInitParam = " + this.myInitParam +"<BR/>");
    response.getWriter().write("myContextParam = " + myContextParam + "</body></html>");
  }
}
```


HWZ Hochschule für Wirtschaft Zürich

Java Servlets: (init parameter using annotations)

```
@WebServlet(
            description = "This Servlet checks the login credentials",
            urlPatterns = { "/Validate" },
            initParams = {
                        @WebInitParam(name="ValidPassword", value="HWZ", description="Test Password")
            }
)
public class Validate extends HttpServlet {
:
}



@WebServlet("/HelloHwzServlet")
public class HelloHwzServlet extends HttpServlet {
:
}
```

Java Servlets: Cookies

```
:
 // Setzen eines Cookies
 Cookie aCookie = new Cookie("myCookie", "myCookieValue");

 aCookie.setValue("myCookieValue");
 aCookie.setComment("Ein Kommentar");


 // cookie wird mitgesendet, an alle requests auf "rothlin.com" und deren sub-domains WICHTIG .

 aCookie.setDomain(" . rothlin.com");


 // Cookie wird mitgesendet auf alle requests "unter" /stat
 aCookie.setPath("/stat");

 aCookie.setMaxAge(seconds);  // if seconds < 0 then Deleted after browser is closed
                              // if seconds = 0 then Cookie is deleted
 response.addCookie(aCookie);


 // Lesen aller Cookies
 Cookie[] cookies = request.getCookies();


:
```

Max. 4kB data

http://docs.oracle.com/javaee/6/api/javax/servlet/http/Cookie.html

Nicht alltägliches Beispiel, was mit Cookies machen kann: http://www.homepage-performance.de/cookies-path.html

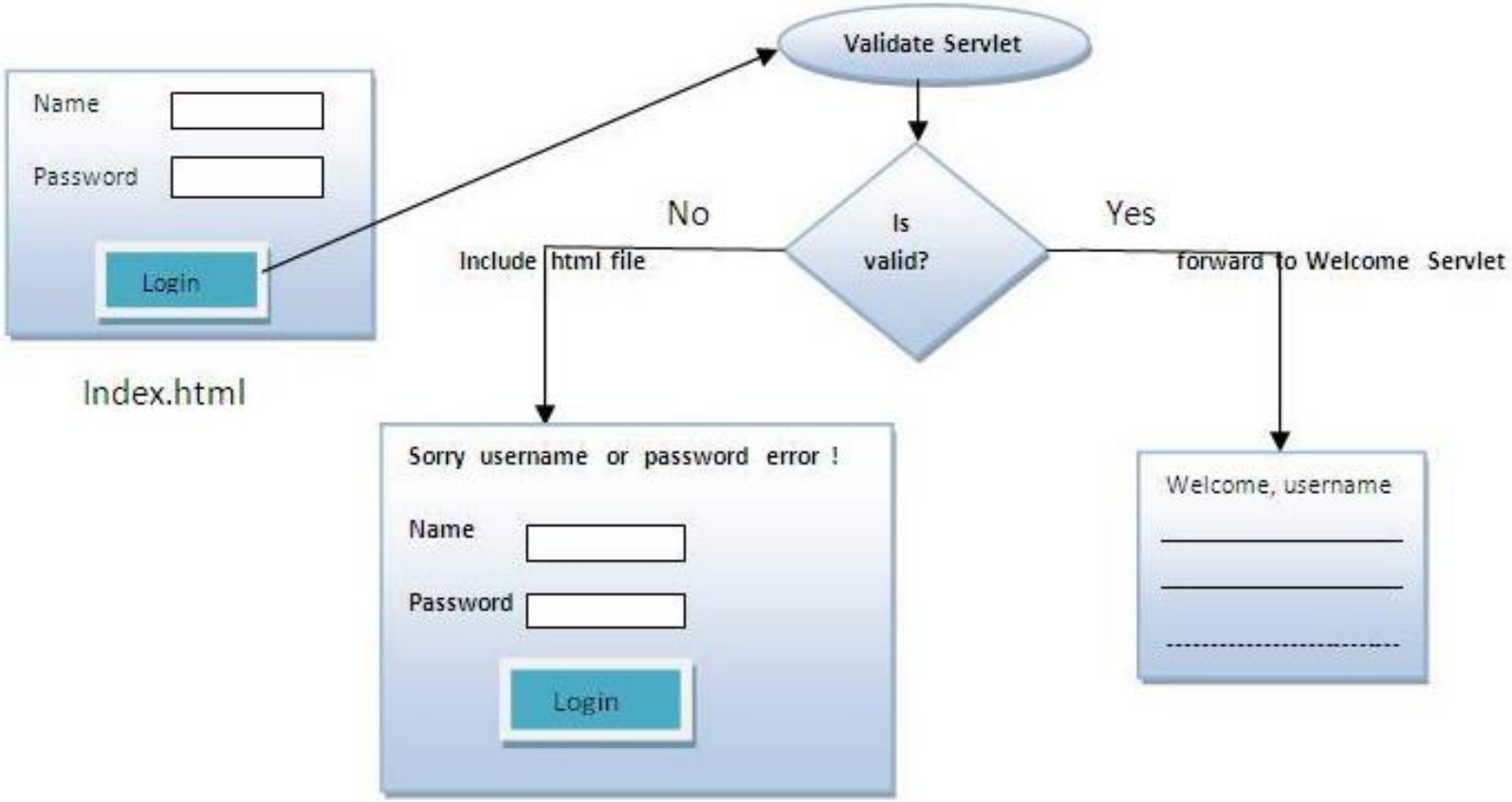
Übung Servlet

- Implementieren Sie ein «`HelloHwzWorld.java`» Servlet.
  `(HWZ_Servlet:SimpleServlet danach HWZ_Servlet:HelloHwzServlet)`
- Implementiere ein «Echo-Servlet». Erstelle eine html –Formular und senden Sie die Formular-Elemente an dieses «Echo-Servlet».
  `(HWZ_Servlet:FormTest.html → FormHandler.java)`
- Untersuchen Sie folgende Speichermöglichkeiten von Daten
  `(HWZ_InitContextParameter:AccessContextInitParam):`
  - Servlet Init-Parameter
  - Context Init-Parameter
  - Context-Variablen
  - Session-Variablen
- Untersuchen Sie in einem konkreten Beispiel den Unterschied von **sendRedirect**, **include** und **forward** (`HWZ_Servlet:RequestDispatcherForwarder`)
- Bauen Sie ein Log-Filter in das HelloWorld Servlet ein
  `(HWZ_ServletFilter:HelloWorldServlet)`

Übung Servlet (2.Teil)

- Erstellen Sie ein neues Projekt `HWZ_LoginServlet`. Entwickeln Sie folgendes System wie unten abgebildet. Für die Validierung des Passwords verwenden Sie einen einfachen Vergleich mit einem hard-codierten Password.

.



Java Servlets: Lösung – Übung (Display Form-Values)

```
protected void processRequest(
                HttpServletRequest request,
                HttpServletResponse response)
                        throws ServletException, IOException {
        response.setContentType("text/html");
        StringBuffer sb = new StringBuffer("<html>\n<head></head>\n<body>\n“);
        String name;
        for (Enumeration<String> e = request.getParameterNames();
                                e.hasMoreElements(); ){
                name = (String)e.nextElement();
                sb.append(name + “: “ + request.getParameter(name) + “<p>“);
        }
        sb.append(“</body>\n</html>\n“);
        response.getWriter().print(sb.toString());
}
```

HWZ_Servlet:FormHandler

Java Servlets: Lösung – Übung (Forwards, includes,….)

```
private void processRequest_Redirect(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {
    response.setHeader("Content-Type", "text/plain");
    response.getWriter().println("Start call: RequestDispatcherForwarder::processRequest_Redirect()");
    // response.sendRedirect("HelloHwzServlet?Application=sendRedirect");
    // response.sendRedirect("http://www.hwz.brunnenhoefli.ch/");
    response.getWriter().println("End call  : RequestDispatcherForwarder::processRequest_Redirect()");
}
private void processRequest_Dispatcher(HttpServletRequest request,HttpServletResponse response)
                                throws ServletException, IOException {
    response.getWriter().println("Start call: RequestDispatcherForwarder::processRequest_Dispatcher()<BR/>");
    RequestDispatcher requestDispatcher = (RequestDispatcher)
          request.getRequestDispatcher("/HelloHwzServlet?Application=RequestDispatcher");
    /* Forward:no merging of response output */
    // ((javax.servlet.RequestDispatcher) requestDispatcher).forward(request, response);
    /* Include: merging of response output */
    ((javax.servlet.RequestDispatcher) requestDispatcher).include(request, response);
    ((javax.servlet.RequestDispatcher) requestDispatcher).include(request, response);
        response.getWriter().println("End call  : RequestDispatcherForwarder::processRequest_Dispatcher()<BR/>");
}
```

HWZ_Servlet:RequestDispatcherForwarder

Java Servlets: Lösung – Übung (Speichermöglichkeiten)

```
:
public void init(ServletConfig servletConfig) throws ServletException {
    super.init(servletConfig);
    myInitParameter      = servletConfig.getInitParameter("myInitParam");
    myContextParameter_2 = getServletContext().getInitParameter("myContextParam_2");
}

protected void processRequest( HttpServletRequest request, HttpServletResponse response)……
    ServletContext context = request.getSession().getServletContext();
    myContextParameter = context.getInitParameter("myContextParam");
    myInitParameter_2  = getInitParameter("myInitParam_2");
    contextParameter = (String) context.getAttribute("aContextVariable");
    :
    :
    context.setAttribute("aContextVariable", myUrlParam);
:
```

HWZ_InitContextParameter:AccessContextInitParam

Java Servlets: Lösung – Übung (Speichermöglichkeiten)

```
:
    <context-param>
      <param-name>myContextParam</param-name> <param-value>myContextParam_Value_1</param-value>
    </context-param>
    <context-param>
      <param-name>myContextParam_2</param-name> <param-value>myContextParam_Value_2</param-value>
    </context-param>

   <servlet>
     <servlet-name>ParameterServlet</servlet-name>
     <servlet-class>com.rothlin.hwz.AccessContextInitParam</servlet-class>
     <init-param>
           <param-name>myInitParam</param-name> <param-value>myInitParam_Value_1</param-value>
     </init-param>
     <init-param>
           <param-name>myInitParam_2</param-name> <param-value>myInitParam_Value_2</param-value>
     </init-param>
   </servlet>
</web-app>
```

HWZ_InitContextParameter:web.xml

Java Servlets: Lösung – Übung (Anmelde-Screens)

```
<form action="Validate" method="get">
   Name:    <input type="text"     name="userName"/><br/>
   Password:<input type="password" name="userPass"/><br/>
   <input type="submit" value="login"/>
</form>
```

HWZ_LoginServlet:index.jsp

```
public class Validate extends HttpServlet {
            public void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {
                        response.setContentType("text/html");
                        PrintWriter out = response.getWriter();
                        String p=request.getParameter("userPass");
                        if(p.equals("HWZ")){
                                    RequestDispatcher rd=request.getRequestDispatcher("Welcome");
                                    rd.forward(request, response);
                        } else {
                                    out.print("Sorry username or password error!");
                                    RequestDispatcher rd=request.getRequestDispatcher("index.jsp");
                                    rd.include(request, response);
                        }
            }
}
```

HWZ_LoginServlet:Validate.java

Java Servlets: Lösung – Übung (Anmelde-Screens)

```
public class Welcome extends HttpServlet {

        public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {

                response.setContentType("text/html");
                PrintWriter out = response.getWriter();

                String name = request.getParameter("userName");
                out.print("Welcome" + name);
        }

}
```

HWZ_LoginServlet:Welcome.java

PHP (Personal Home Page)

PHP: Hypertext Preprocessor (Personal Home Page)

http://www.w3schools.com/php/default.asp

```
<html>
    <body>

    <?php
    // this are commands executed on the server for each request to this page
    echo "Hello World";


    $a = 5;   // global scope
    $b = 10;  // global scope

    function myTest($initValue) {
           $localVariable = $initValue * 2                // Locale Variable
           static $aStaticVariable  // keeps the value to the next call
           global $a;                          // 1.Möglichkeit für Zugriff auf globale Variablen
           $GLOBALS['b'] = $a + $GLOBALS['b']; // 2.Möglichkeit für Zugriff auf globale Variablen
    }

    myTest(3);
    echo $b;
    ?>

    </body>
</html>
```

PHP: Form value access

```
<form action="welcome.php" method="get">
     Name: <input type="text" name="fname">
     Age:  <input type="text" name="age">
     <input type="submit">
</form>

welcome.php
Welcome <?php echo $_GET["fname"]; ?>.<br>
You are <?php echo $_GET["age"]; ?> years old.
```

Get: Parameter als URL Parameter

```
<form action="welcome.php" method="post">
     Name: <input type="text" name="fname">
     Age:  <input type="text" name="age">
     <input type="submit">
</form>

welcome.php
Welcome <?php echo $_POST["fname"]; ?>.<br>
You are <?php echo $_POST["age"]; ?> years old!
```

POST: Parameter in Message Body

_REQUEST: Zugriff auf Parameter

unabhängig ob mit GET oder POST gesendet

(Enthält auch _COOKIE[] Werte!)

PHP: File Zugriff

```
<?php
    $file=fopen("welcome.txt","r") or exit("Unable to open file!");
    while(!feof($file)) {
      echo fgets($file). "<br>";   // read line by line // fwrite() for writing to a file
    }
fclose($file);
?>
```

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

PHP: Cookies

```
<?php
   //Setzen eines Cookies
   //Expires after 3600s setcookie(name, value, expire, path, domain);
   setcookie("user", "Alex Porter", time()+3600);

   // Lesen des Cookies
   if (isset($_COOKIE["user"]))
     echo "Welcome " . $_COOKIE["user"] . "!<br>";
   else
     echo "Welcome guest!<br>";

   // Löschen eines Cookies --> set the expiration date to one hour ago
   setcookie("user", "", time()-3600);
?>
```

PHP: Sessions

```
<?php
  session_start();

  if(isset($_SESSION['views']))
     $_SESSION['views']=$_SESSION['views']+1;
  else
     $_SESSION['views']=1;
  echo "Views=". $_SESSION['views'];
?>
```

JSP (Java Server Page)

JSP: Java Server Page (First Example)

```
<%@ page import="java.util.*" %>
<html>
  <body>
    <%    // This is a scriptlet (java code! Therefore comments java-sytle.)
          System.out.println( "Evaluating date now" ); // goes to server log
          Date date = new Date();                      // java.util nicht nötig, da page import
          out.println("1: " + String.valueOf( date )); // goes to the HTML stream
    %>
    <BR/>Hello!  The time is now <%= date %> or <BR/>   <!--This is an expression -->
    <%
          out.println( date );
          out.println( "<BR>Your machine's address is " );
          out.println( request.getRemoteHost());
    %>
  </body>
</html>
```

HWZ_JSP_Bean:index.jsp

JSP-tutorial: http://www.jsptut.com/

JSP: Mixing scriplets and HTML

```
<TABLE BORDER=2>
   <%  for ( int i = 0; i < n; i++ ) { %>
          <TR><TD>Number</TD><TD><%= i+1 %></TD></TR>
   <%  } %>
</TABLE>

<%  if ( hello ) {   %>
        <P>Hello, world
<%  } else {         %>
        <P>Goodbye, world
<%  } %>
```


HWZ Hochschule für Wirtschaft Zürich

JSP: Declaration

```
<%@ page import="java.util.*" %>
<HTML> <BODY>
<%!
     //Variable will be set only once the page is loaded (into container)
     Date theDate = new Date();
     Date getDate()     {
         System.out.println( "In getDate() method" );
         return theDate;
     }
%>
Hello!  The time is now <%= getDate() %>
</BODY> </HTML>
```

JSP: JSP Tags

```
<HTML>
   <BODY>
      Going to include hello.jsp...<BR>
      <jsp:include page="hello.jsp"/>
   </BODY>
</HTML>
```

JSP: Sessions

GetName.html

```
<HTML> <BODY>
 <FORM METHOD=POST ACTION="SaveName.jsp">
   What's your name?
   <INPUT TYPE=TEXT NAME=username SIZE=20> <P>
   <INPUT TYPE=SUBMIT>
 </FORM>
</BODY> </HTML>
```

SaveName.jsp

Wert in Session Hash speichern

```
<%
    String name = request.getParameter( "username" );
    session.setAttribute( "theName", name );
%>
<HTML> <BODY>
   <A HREF="NextPage.jsp">Continue</A>
</BODY> </HTML>
```

NextPage.jsp

Wert von Session Hash lesen

```
<HTML> <BODY>
   Hello, <%= session.getAttribute( "theName" ) %>
 </BODY> </HTML>
```

JSP: Beans and Form processing

GetName.html

```
<HTML> <BODY> <FORM METHOD=POST ACTION="NextPage.jsp">
 What's your name? <INPUT TYPE=TEXT NAME=username SIZE=20><BR>
 What's your e-mail address? <INPUT TYPE=TEXT NAME=email SIZE=20><BR>
 What's your age? <INPUT TYPE=TEXT NAME=age SIZE=4> <P>
 <INPUT TYPE=SUBMIT>
</FORM> </BODY> </HTML>
```

page: nur in diesem Servlet in diesem Request
request: in diesem Request über alle beteiligten Servlet
session: während der Session über alle Servlets
application: für alle sessions bis Servlet neu geladen wird

NextPage.jsp

```
<jsp:useBean id="userBean" class="user.UserData" scope="session"/>
<jsp:setProperty name="userBean" property="*"/>  <!- Property-binding ->
<HTML> <BODY> You entered<BR>
Name: <%= userBean.getUsername() %><BR>
Email: <%= userBean.getEmail() %><BR>
Age: <%= userBean.getAge() %><BR>
 </BODY> </HTML>
```

UserData.java

```
package user;
public class UserData {
    String username;
    String email;
    int    age;
    public void   setUsername( String value )   { username = value; }
    public void   setEmail( String value )      { email    = value; }
    public void   setAge( int value )           { age      = value; }
    public String getUsername()                 { return username;  }
    public String getEmail()                    { return email;     }
    public int    getAge()                      { return age;       }
}
```

Bean class: Needs to be placed in Web-Servers Class-Path under sub dir user

`userBean` ist Name der Bean

HWZ Hochschule für Wirtschaft Zürich

JSP: Beans and Form processing

jsp:setProperty

```
<jsp:setProperty name="myBean" property="foo" value="bar" />
```

entspricht dem Java-Code:

```
myBean.setFoo("bar");
```

Bei der Verarbeitung eines HTTP-Requests einsetzbar für das Setzen einer einzelnen Property

```
<jsp:setProperty name="myBean" property="foo" param= "foo_val"
/>
```

entspricht dem Java-Code:

```
myBean.setFoo(request.getParameter("foo_val"));
```

JSP: Beans and Form processing

jsp:setProperty

Bei der Verarbeitung eines HTTP-Requests einsetzbar für das Setzen **aller** Properties

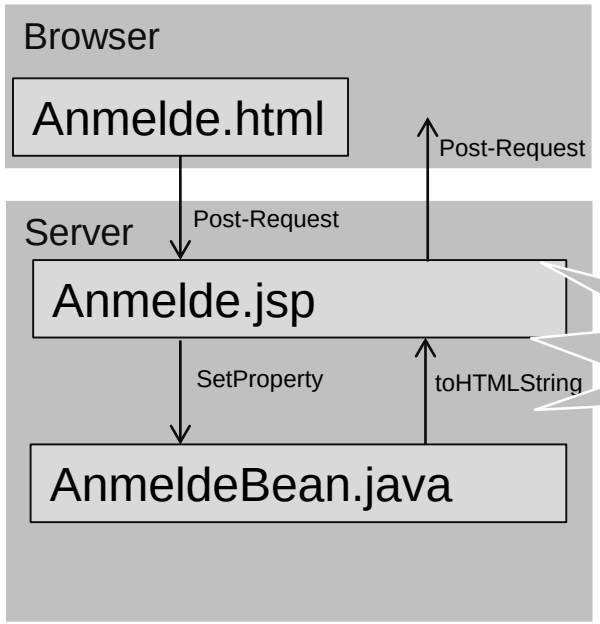
```
<jsp:setProperty name="myBean" property="*"/>
```

entspricht dem Java-Code:

```
myBean.set<req-par1>(request.getParameter(<req-par1>));
myBean.set<req-par2>(request.getParameter(<req-par2>));
```

Bei der Zuweisung wird eine automatische Typenkonvertierung vorgenommen!!

JSP: Beans and Form processing



Setter / Getter müssen mit Feldnamen im Form (Anmelde.htm) übereinstimmen!

```
package com.rothlin.hwz;
@Stateless
@LocalBean
public class AnmeldeBean {
…
    /* Default Constructor wichtig!! */
    public AnmeldeBean() {…}

    /* Setter und Getter Methoden */
    public void setName(string name) {…}
    public String getName() {…}
    …
    /* HTML String */
    public toHTMLString() {…}
}
```

Anmelde.java

```
<html><head><title>Anmeldung</title></head>
<body>
   <%@page import= "com.rothlin.hwz.AnmeldeBean" %>
   <jsp:useBean id="anmeldung" class="com.rothlin.hwz.AnmeldeBean" >
           <jsp:setProperty name="anmeldung" property="*" />
   </jsp:useBean>
   <%= anmeldung.toHTMLString() %>
</body>
</html>
```

Anmelde.jsp

HWZ Hochschule für Wirtschaft Zürich

JSP: Controllers

NextPage.jsp

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>
<jsp:setProperty name="user" property="*"/>
<%
 String tgtPage = null;
 if ( user.choice.equals( "choice1" ))
        tgtPage = "tgt1.jsp";
 else if ( user.choice.equals( "choice2" ))
        tgtPage = "tgt2.jsp";
 else
        tgtPage = "tgtDefault.jsp";

 response.sendRedirect( tgtPage );
%>
```

This JSP only checks (pre-processing, validates) form values, stores the values in the bean and directs it to the next JSP

JSP: Tags (Übersicht)

```
<%= Expression %>
<% Code / Scriptlet %>
<%! Declaration %>
<%-- Comment --%>
<%@ Directive %>
        <%@ include file     = "relative URL" %>
        <%@ page import      = "java.util.*" %>
        <%@ page contentType = "text/html" %>
```

XML Schreibweise

```
<jsp:expression>Expression</jsp:expression>
<jsp:scriptlet>Code</jsp:scriptlet>
<jsp:directive.xxx attribute= "value" />
        <jsp:directive.include file     = " relative URL " />
        <jsp:directive.page import      = "java.util.*" />
        <jsp:directive.page contentType = "text/html" />
```

JSP: Vordefinierte Variablen (Übersicht)

HttpServletRequest request;

Der aktuelle abgearbeitete HttpServlet-Request

HttpServletResponse response;

Die Response für diesen HttpServlet-Request

PrintWriter out;

Ausgabe-Strom für Antwort

HttpSession session;

Daten der aktuellen Benutzersitzung

ServletContext application;

Application-Context

ServletConfig config;

Servlet-Konfiguration

PageContext pageContext;

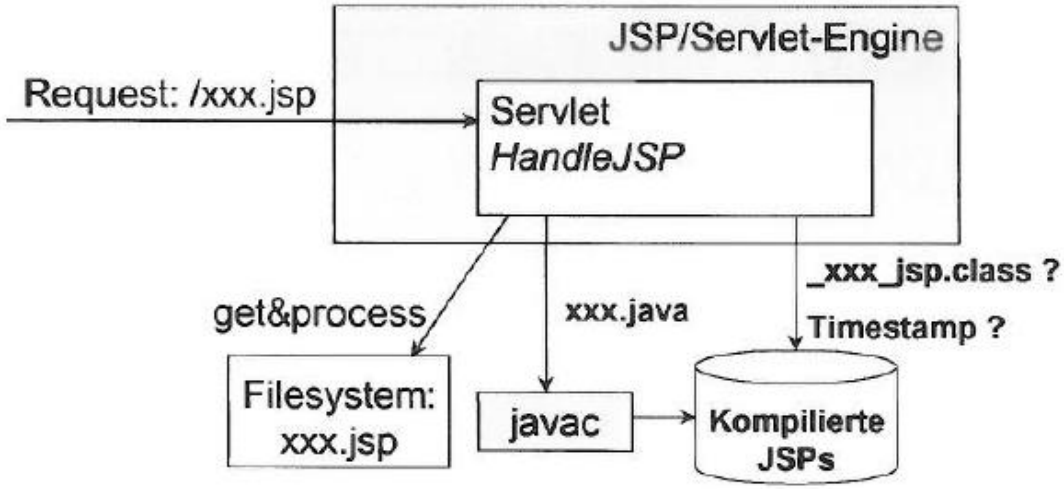
Seiten-Kontext

Exception exception;

Exception-Object, falls ein Fehler ausgelöst wurde

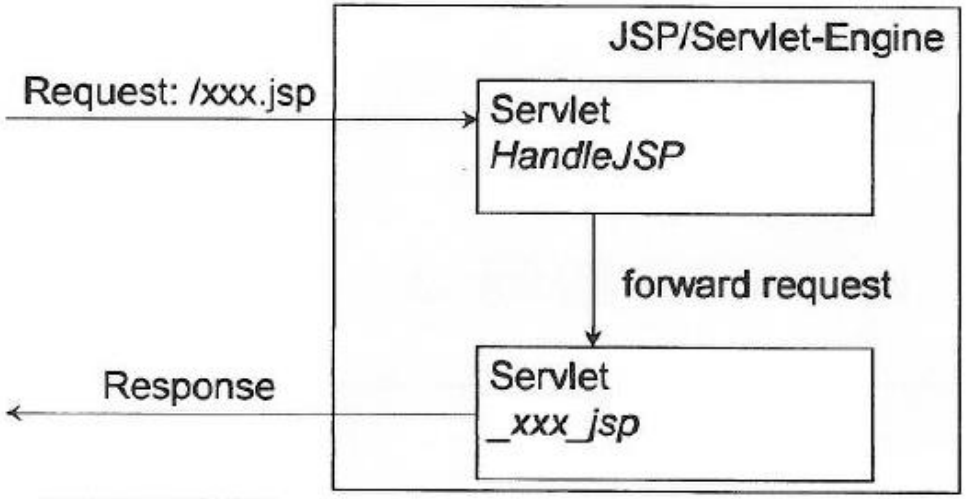
JSP: Is compiled to a servlet

1. Request nach Update von xxx.jsp



First time the jsp-Page is compiled to a servlet

2. Weitere Requests



jsp-Page is forwarding the Request to the servlet

HWZ Hochschule für Wirtschaft Zürich

JSP: Generated Servlet

Das Servlet enthält _jspService() Methode

- Definiert alle impliziten Variablen
- Setzt HTML-Code in out.write("<HTML-Content>")
- Setzt Java-Code (<%....%>) direkt ein
- Setzt Expression (<%=expr%>) in die Methode out.write(expr) ein
- Enthält Instanzierungscode für <jsp:useBean> Konstrukte

JSP: Exception Handling

Not via Catch-block ➔ Declare Error-Page

```
<%@ page errorPage="/jsp/ErrorPage.jsp" %>
<html><head></head><body>
Fehlertest: <%= 12/0 %>
</body></html>
```

index.jsp

```
<%@ page isErrorPage="true" %>
<html><head></head><body>
  <% if (exception != null) { %>
        Fehler: <%= exception.getMessage()  %>
  <% } else { %>
        Unbekannter Fehler!
   <% } %>
</body></html>
```

ErrorPage.jsp

JSP: Java Standard Tag Library (JSTL)/Expression Language

Tags anstelle Java-Code in JSPs → Tag Libraries für wiederkehrende Aufgaben wurden Standardisiert (JSTL)

JSTL: http://www.jsptutorial.org/content/jstl

JSTL-Core: http://www.jsptutorial.org/content/jstl_core

Expression Language ersetzt scriptlet (JSP bleibt lesbarer)

EL: http://www.jsptutorial.org/content/el

```
<c:forEach var="someWord" items="${someListOfStrings}">
   <c:out value="${someWord}" />
</c:forEach>
```

JSTL 1.0

```
<c:forEach var="someWord" items="${someListOfStrings}">
   ${someWord}
</c:forEach>
```

JSP 2.0

JSP: Java Standard Tag Library (JSTL)/Expression Language

JSTL's fünf Tag-Libraries

Name	prefix	URI	Aufgabe
core	c	http://java.sun.com/jsp/jstl/core	Basistags, die vor allem Programmablauf-Tags wie Schleifenkonstrukte oder Verzweigungen enthalten
fmt	fmt	http://java.sun.com/jsp/jstl/fmt	Tags zur Formatierung von Datums-, Währungs- und Zahlwerten und zur Internationalisierung
xml	x	http://java.sun.com/jsp/jstl/xml	Geeignet zur Bearbeitung von XML-Dokumenten. Enthält u.a. wie core Schleifen- und Verzweigungstags. Der Unterschied zu den core-Tags liegt darin, dass XPath-Ausdrücke zur Bedingungsentscheidung bzw. zur Feststellung der Menge dienen, über die iteriert werden soll
sql	sql	http://java.sun.com/jsp/jstl/sql	Tag-Bibliothek zum Ansprechen einer Datenbank direkt aus der JSP heraus
functions	fn	http://java.sun.com/jsp/jstl/functions	Die Tags dieser Bibliothek wenden Funktionen auf Expression Language-Ausdrücke an. Es handelt sich mit einer Ausnahme durchweg um String-Manipulationen

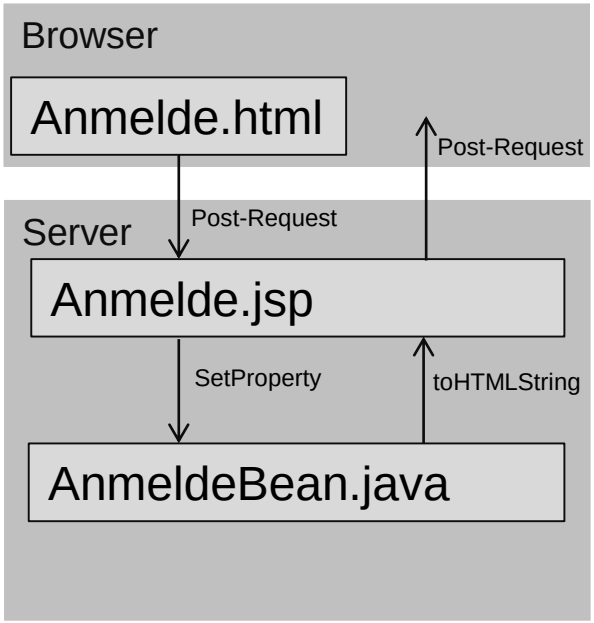
Einbinden: `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`

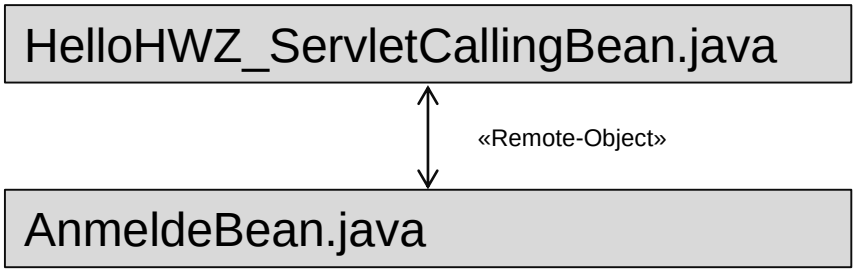
Übung JSP

- Implementieren Sie ein «HelloWorld» JSP. Was ist der Unterschied zu einer normalen HTML-Page? `(HWZ_Servlet:index.jsp)`
- Fügen Sie zum helloWorld.jsp noch scriplets hinzu (z.B. Datum / Zeit, Tabelle mit 2er-Potenzen,… auf der Seite anzeigen) `(HWZ_JSP_Bean:index.jsp)`
- Erstellen Sie eine Error-Page und deklarieren Sie in ihrem helloWorld.jsp diese Seite. Bauen Sie einen Fehler (z.B. Div by 0 in einer JSP-Expression) ein, welcher durch einen URL-Parameter `?Error=True` aktiviert werden kann und testen Sie das Fehlerverhalten. `(HWZ_JSP_Bean:index.jsp)`

Übung JSP

- Implementiere eine Anmelde-Applikation, bestehend aus einem einfachen HTML Formular, beim Submit werden die Feldinhalte an eine JSP gesendet, welche die Form-Values in einem Bean speichert. Mit Hilfe einer toHTMLString() Methode in der Bean, werden die gespeicherten Werte im Browser als Antwort angezeigt. Machen Sie ein neues «Dynamisches Web-Projekt» dafür. `(HWZ_JSP_Bean:Anmelde.html)`
- Entwickeln Sie ein Servlet `(HWZ_JSP_Bean:HelloHWZ_ServletCallingBean.java)` welches auf das `AnmeldeBean.java` von verheriger aufgabe zugreift.





Distributed Systems / Distributed Computing

Ziele

- Jeder kennt die Unterschiede von…
  - Socket-Communication
  - Remote Procedure Call (RPC)
  - Remote Method Invocation (RMI)
  - Enterprise Java Beans (EJB)
  - Web-Services (JAX-WS)
  - Representational State Transfer Services (REST)

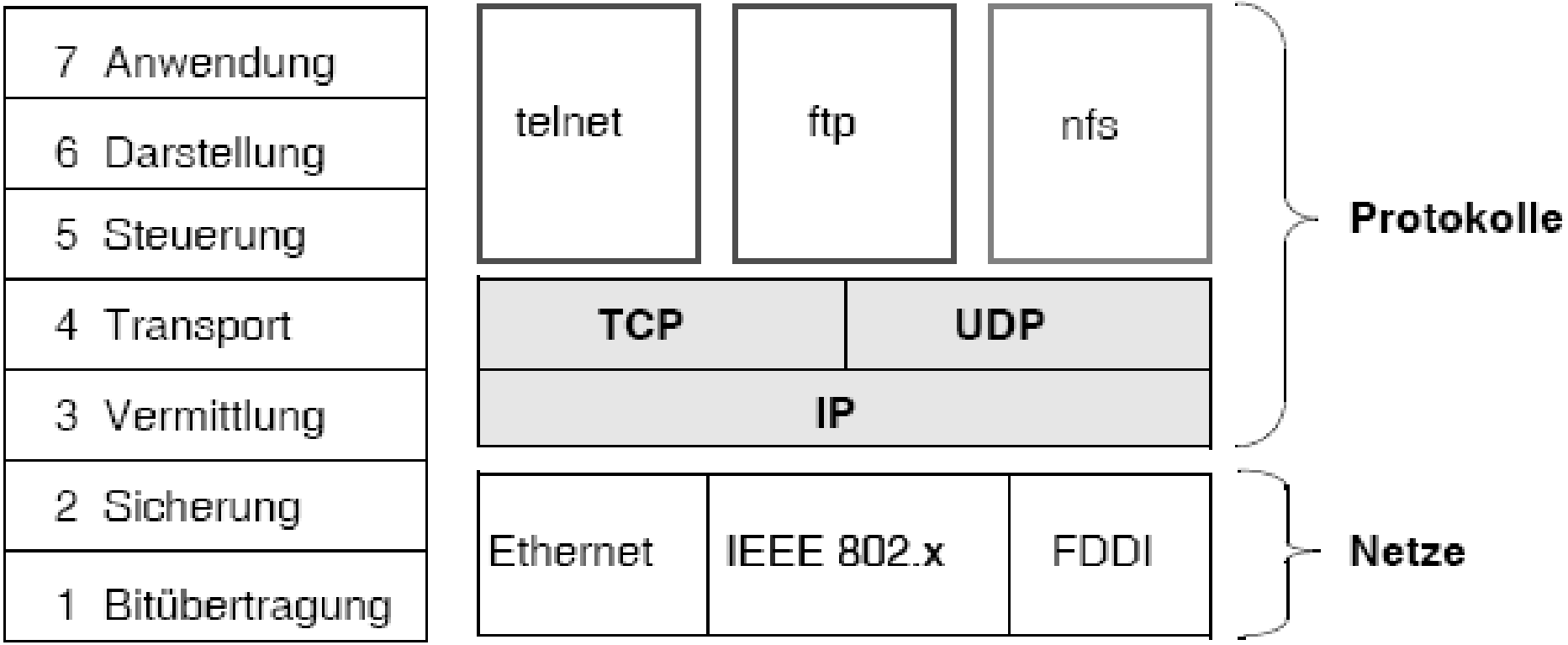
- Jeder hat selber eine lauffähige Applikation mit allen oben genannten Technologien implementiert

- Jeder kann einen öffentlichen Web-Service in einer eigenen Applikation nutzen

Repetitionsfragen / Erstellen Sie einen OnePager

- Was ist eine TCP/IP Message?
- Was ist die Idee hinter dem OSI-Modell?
- Stellen Sie Socket-Communication, RPC, RMI, EJB einander gegenüber?
- Was sind die Kern-Aussagen zu Socket-Communication?
- Was sind die Kern-Aussagen zu RCP?
- Was sind die Kern-Aussagen zu RMI?
- Was sind die Kern-Aussagen zu EJB?
- Wieso stub/skeleton?
- Unterschied synchron/asynchroner Aufruf?

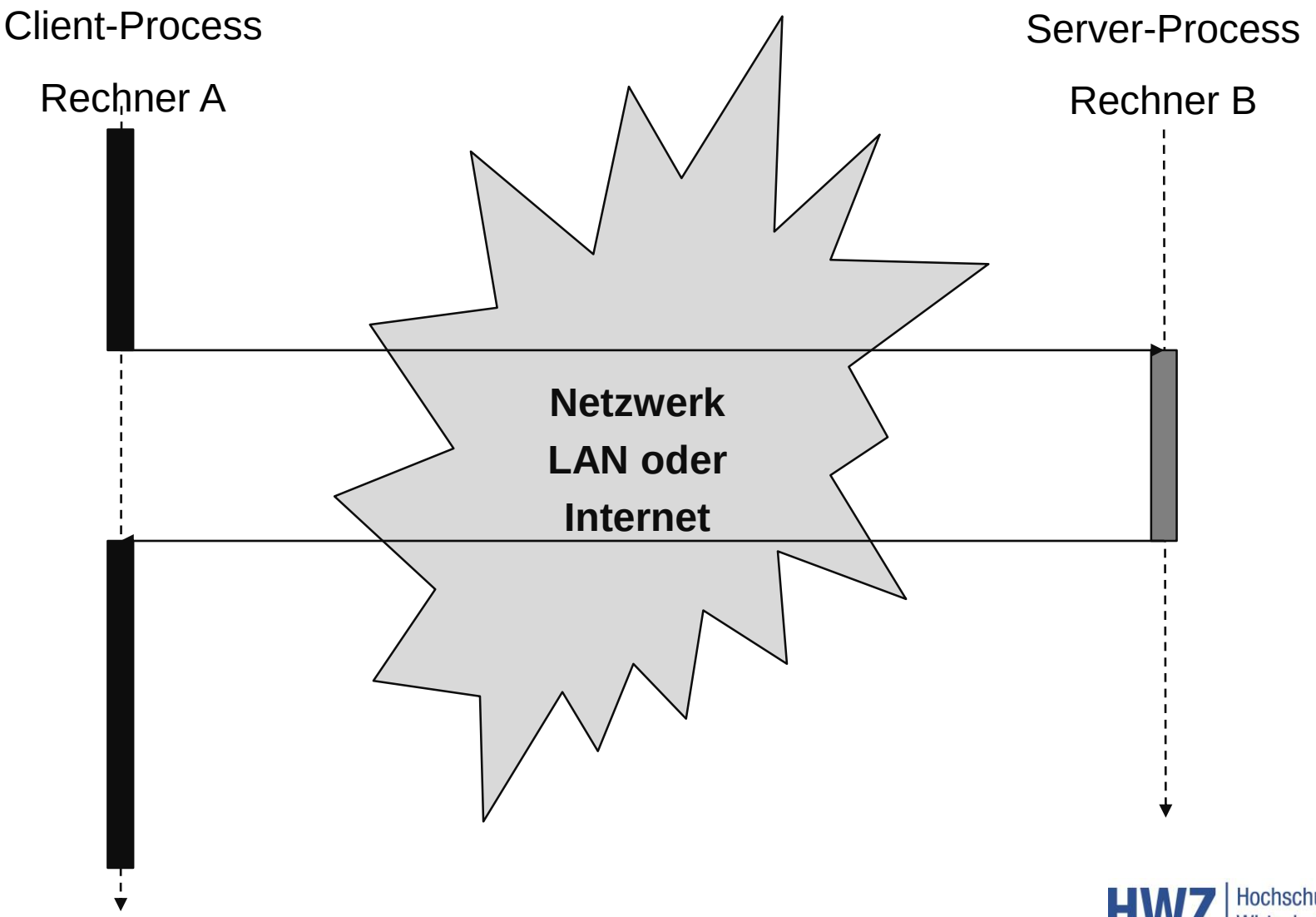
TCP/IP



TCP/IP: Transmission Control Protocol/Internet Protocol

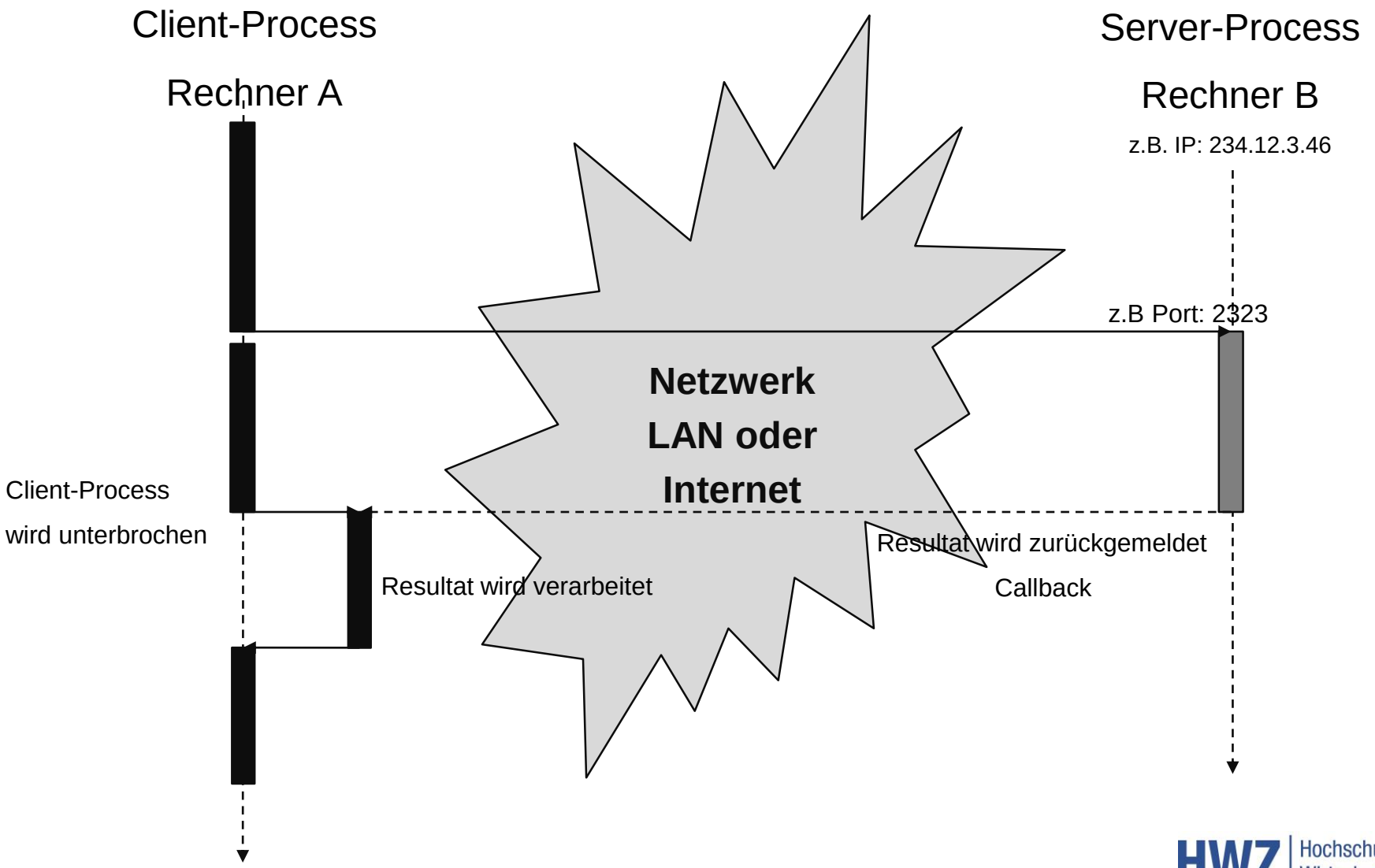
UDP: User Datagram Protocol

TCP/IP (synchroner Aufruf)



HWZ Hochschule für Wirtschaft Zürich

TCP/IP (asynchroner Aufruf)

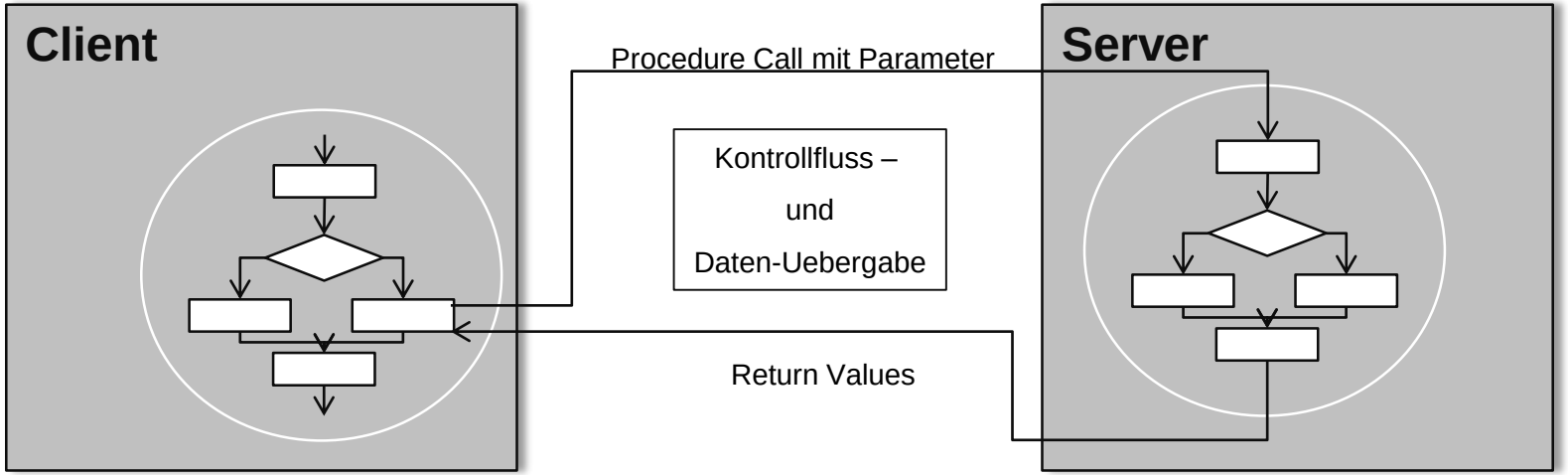


HWZ Hochschule für Wirtschaft Zürich

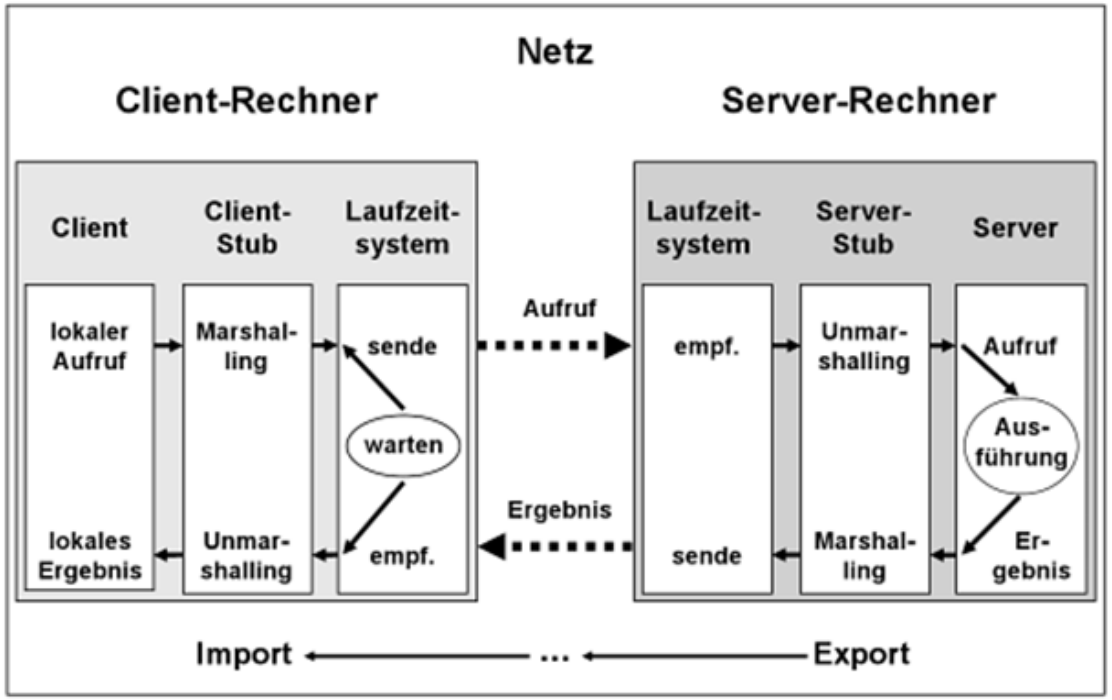
Systemarchitekturen

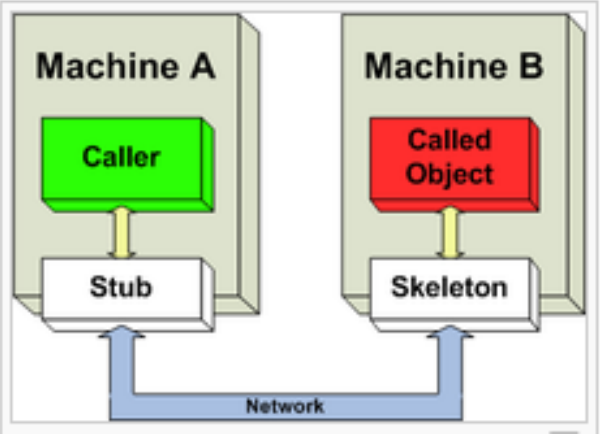
Client-Server:

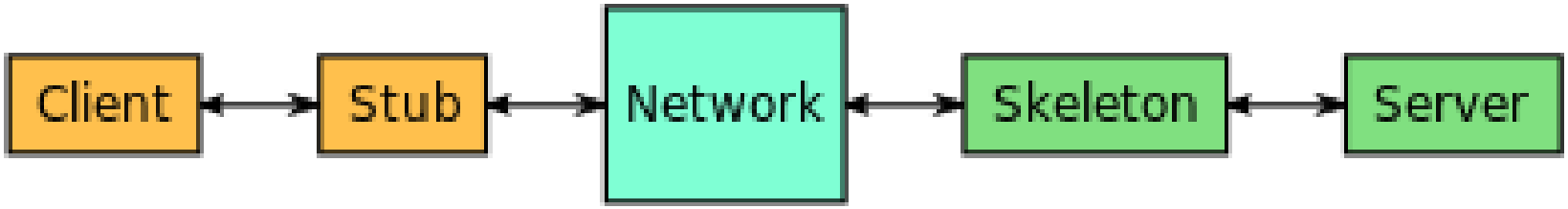
- Entfernte Prozeduraufrufe z.B. Remote Procedure Call (RPC)
- Grundsätzlich synchron (kein Ausnutzen von Parallelität)
  - ➔ Erweitert mit asynchronen Mechanismen (z.B. Callbacks)
- Hindernisse:
  - Datenflüssen / Konvertierung / Andere Adressräume / Andere OS
  - Server-Prozess muss aktiviert werden



RMI (Remote Method Invocation)



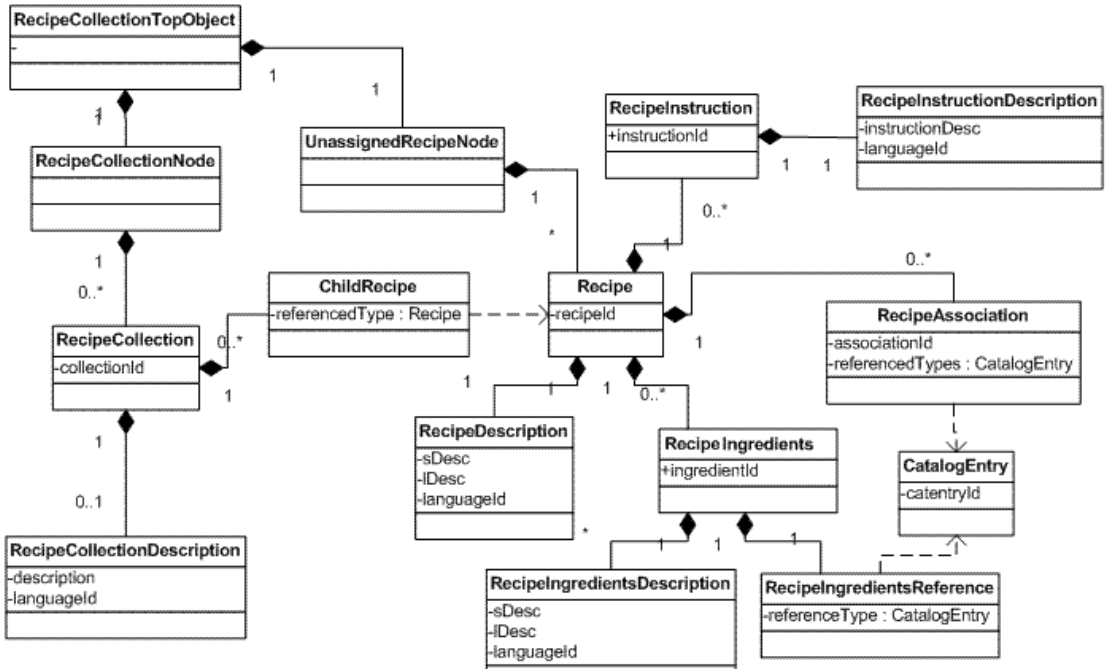




Systemarchitekturen

Distributed Objects:

- Objekte sind verteilt (z.B. RMI (Java world), CORBA (Heterogen),..)
  - Methoden von «fernen Objekten» können aufgerufen werden
  - Objekte können als Parameter Methoden übergeben / return values werden

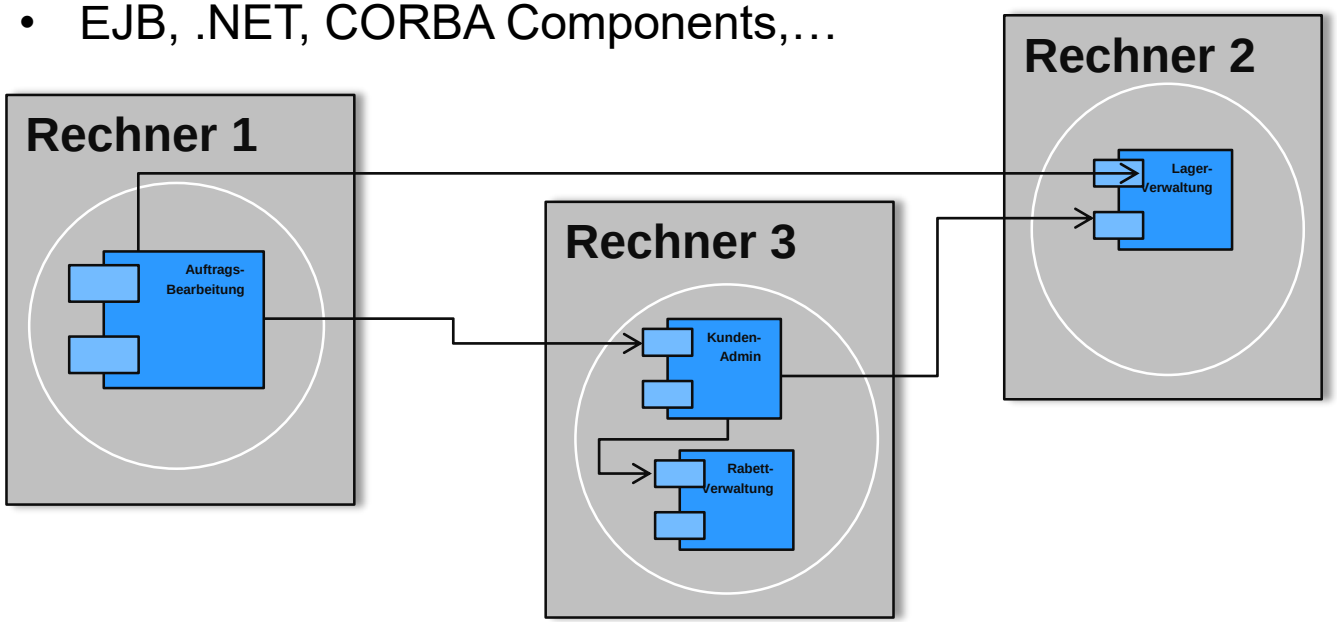


- Hindernisse:
  - Die Verteilung passiert während deployment oder zur Laufzeit
  - Unsichtbar für den Entwickler?
  - Objekte müssen gefunden werden (z.B. JNDI, Registry,IOR..)

Systemarchitekturen

Component Based:

- Component erweitern Objektenmodell
  - Übergeordnete Business-Aufgabe
  - Allgemeine Aufgaben z.B. Persistenz, Transaktions,.. müssen nicht von der Komponente gelöst werden
  - EJB, .NET, CORBA Components,…



Systemarchitekturen

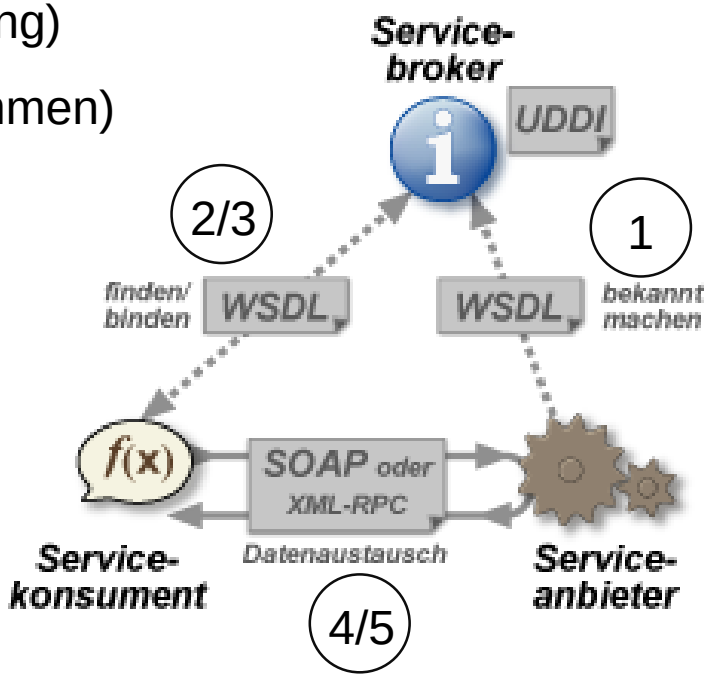
Service Oriented Architectures (SOA) (Web-Services ist ein Spezialfall)

- Grobgranulare Bausteine werden angeboten, gesucht und genutzt
- Wohldefinierte Schnittstelle
- Hohe Verfügbarkeit
- Komposition möglich
  - Orchestrierung: Aus Diensten wird neuer Dienst zusammengesetzt
  - Choreographie: ein Geschäftsprozess wird aus Diensten kombiniert
- Kommunikationsprotokoll: SOAP (Simple Object Access Protokoll)
- Beschreibung des Services (plattform-, sparch- und protokoll-unabhängig):

  WSDL (Web-Service Definition Language)

Web-Services nutzen

1. Veröffentlichen / Bekannt machen in einem Verzeichnis
2. Service suchen (abfragen)
3. Verweis auf Dienst bekommen
4. Details über Dienst abfragen (Beschreibung)
5. Dienst nutzen (Aufruf und Resultat bekommen)



SOA (Service Oriented Architekture)

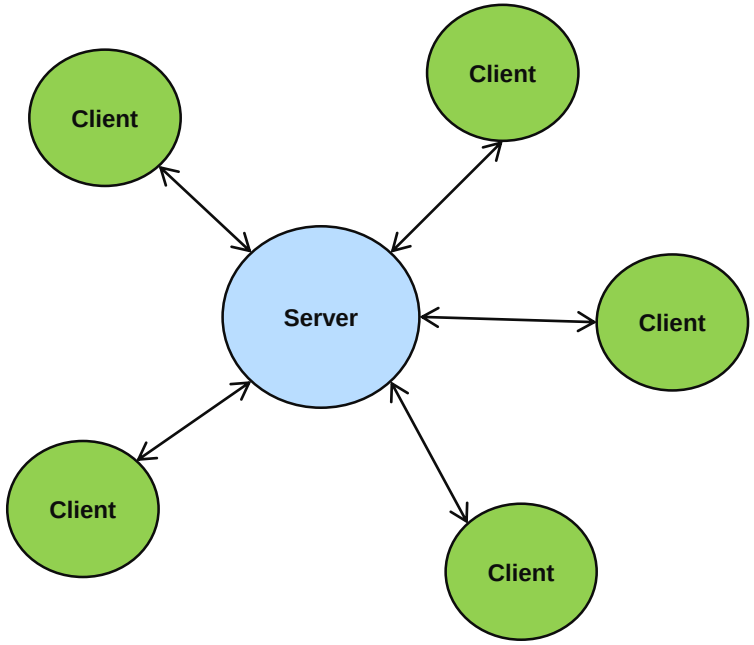
SOAP (Simple Object Access Protokoll)

UDDI (Universal Description, Discovery and Integration)

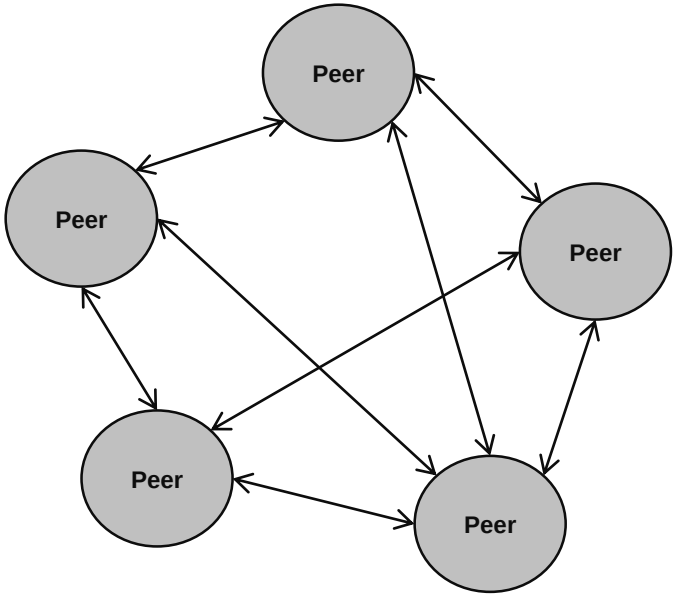
(siehe www.uddi.org)

System Topologie

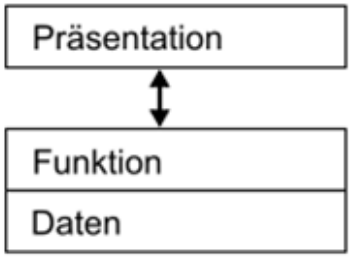
Client-Server (Master – Slave)



Peer to Peer (Gleichberechtigte Partner)

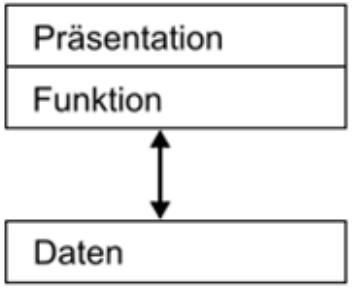


N-Tier (Ebene)



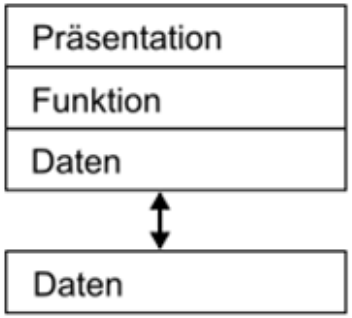
THIN-CLIENT, AKTIVER SERVER
+ zentrale Administration & Wartung
+ Kostenersparnis
+ Sicherheit (nur 1 Server muss geschützt werden)
+ Flexibilität

- hohe Belastung für Server



DATEN-SERVER
→ Server liefert nur benötigte Daten
+ jeder Client kann selber rechnen → höhere Performance
+ zentrale Datenhaltung bleibt bestehen → Sicherheit

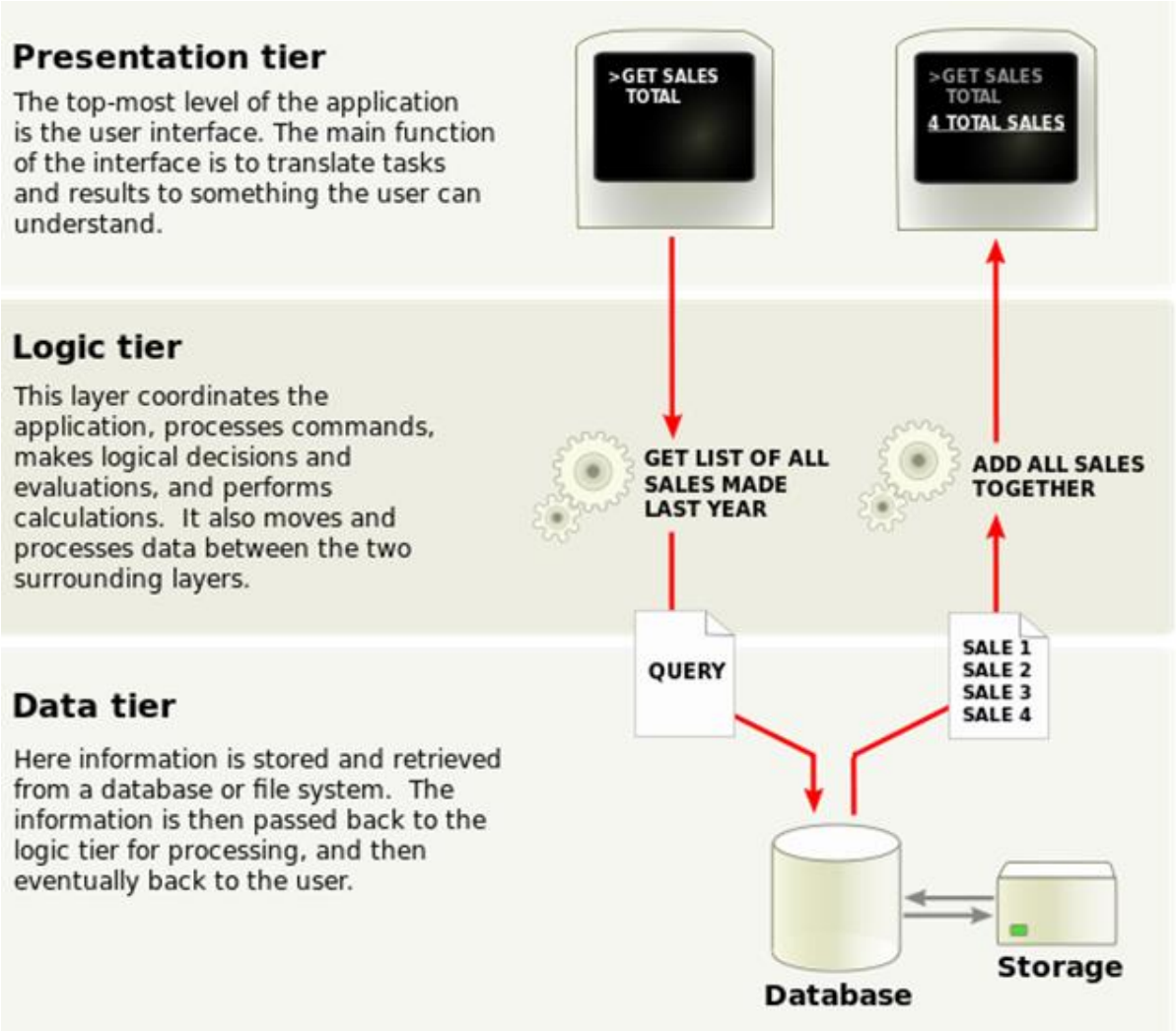
- hoher Installationsaufwand, da jeder Client alle Applikationen braucht



FAT-CLIENT
+ Entlastung Server & Datenleitung durch Plug-Ins
+ Weiterarbeiten möglich, wenn Kommunikation zum Server gestört

- lange Dauer, bis alle Clients Updates haben
- hohe Wartungskosten

N-Tier (Ebene)



business logic
logic tier
data access tier
middle tier

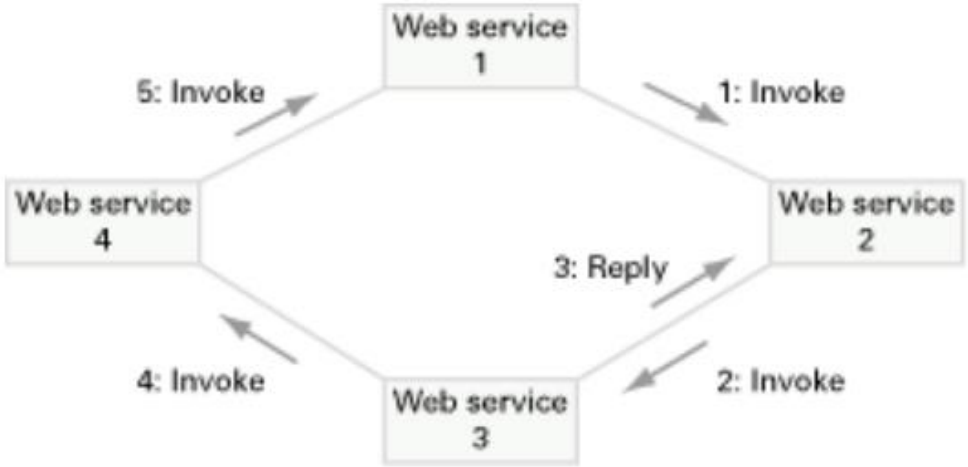
Begriffe: Distributed Computing

Abrev	Begriff
SOA	Service Oriented Architecture
OMA	Object Management Architecture
CORBA	Common Object Request Broker Architecture
IOR	Interoperable Object Reference (eindeutige Object Reference im Internet)
IIOP	Internet Inter-Objectbroker Protokoll
SOAP	Simple Object Access Protokoll
RMI	Remote Methode Invocation (Stubs / Skeleton ; JAVA)
IDL	Interface Definition Language (Sprach Unabhängig) ➔ IDL Compiler

Web-Service kombinieren (Composition of Web services)

Orchestration

Choreography



Composition using BPEL or BPEL4WS

Business Process Execution Language for Web Services

- BPEL ist eine Sprache zum definieren und ausführen von Geschäftsprozessen
- Beschreibt, wie welche Web-Services aufgerufen werden müssen
- Diese Beschreibung erzeugt eine neue Composition
- BPEL Prozesse beinhalten mehrere Schritte
- Diese Schritte nennt man “activity“

BPEL Activities

Primitive activities:

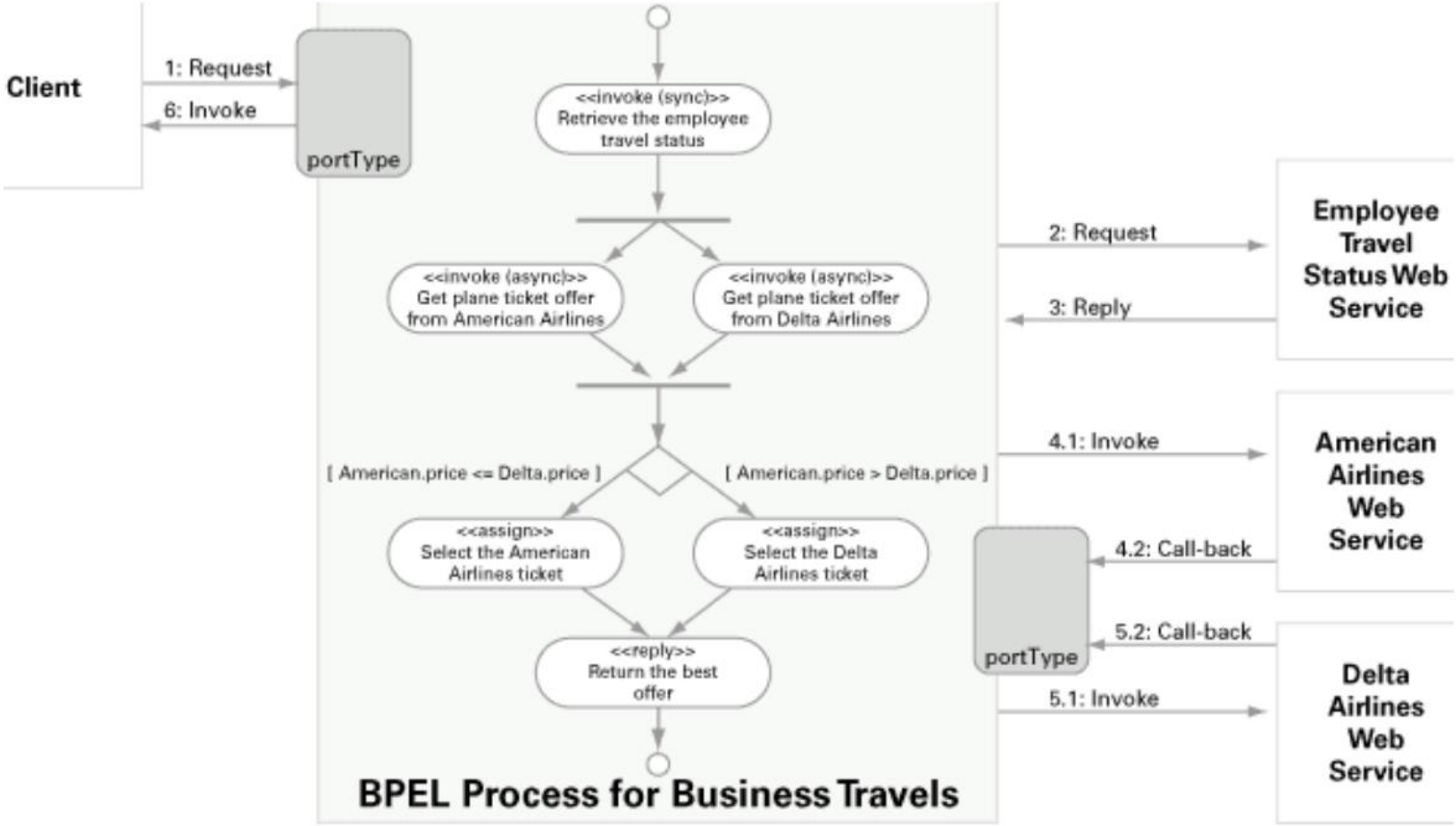
- Invoking other Web services, using `<invoke>`
- Waiting for the client to invoke the business process by sending a message, using `<receive>` (receiving a request)
- Generating a response for synchronous operations, using `<reply>`
- Manipulating data variables, using `<assign>`
- Indicating faults and exceptions, using `<throw>`
- Waiting for some time, using `<wait>`
- Terminating the entire process, using `<terminate>`

BPEL Activities

structure activities:

- Sequence `<sequence>` which allows us definition of a set of activities that will be invoked in an ordered sequence
- Flow `<flow>` for defining a set of activities that will be invoked in parallel
- Case-switch construct `<switch>` for implementing branches
- While `<while>` for defining loops
- The ability to select one of several alternative paths, using `<pick>`

BPEL Example



BPEL Example (steps)

1. Inventory the Involved Web Services
2. Define WSDL for the BPEL Process
3. Define Partner Link Types (for invoked WS as well client)
4. Create the Business Process
5. Deploy and Test

Java Enterprise Architecture J2EE

Containers / Application Server

Java Plattform, Enterprise Edition (J2EE oder Java EE)

- Spezifikation einer Softwarearchitektur (für Java) um verteilte, mehrschichtige und skalierbare Anwendungen zu entwickeln.

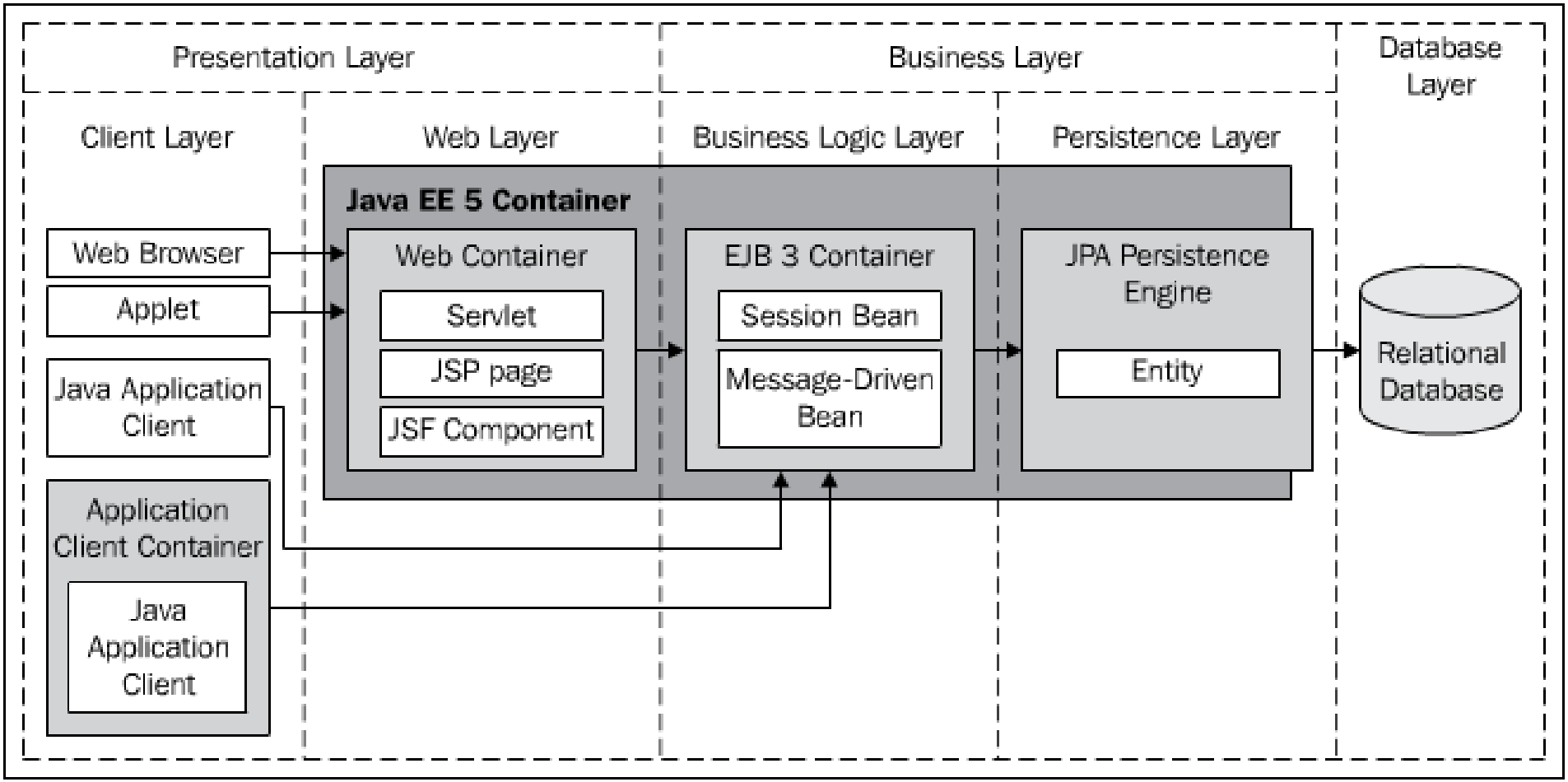
- J2EE Komponenten brauchen eine Laufzeitumgebung (Container). Dieser Container ist Teil eines Application Servers mit klar definierten Funktionen und Schnittstellen.

- .Net- Framework und Plattform Pendant von Microsoft

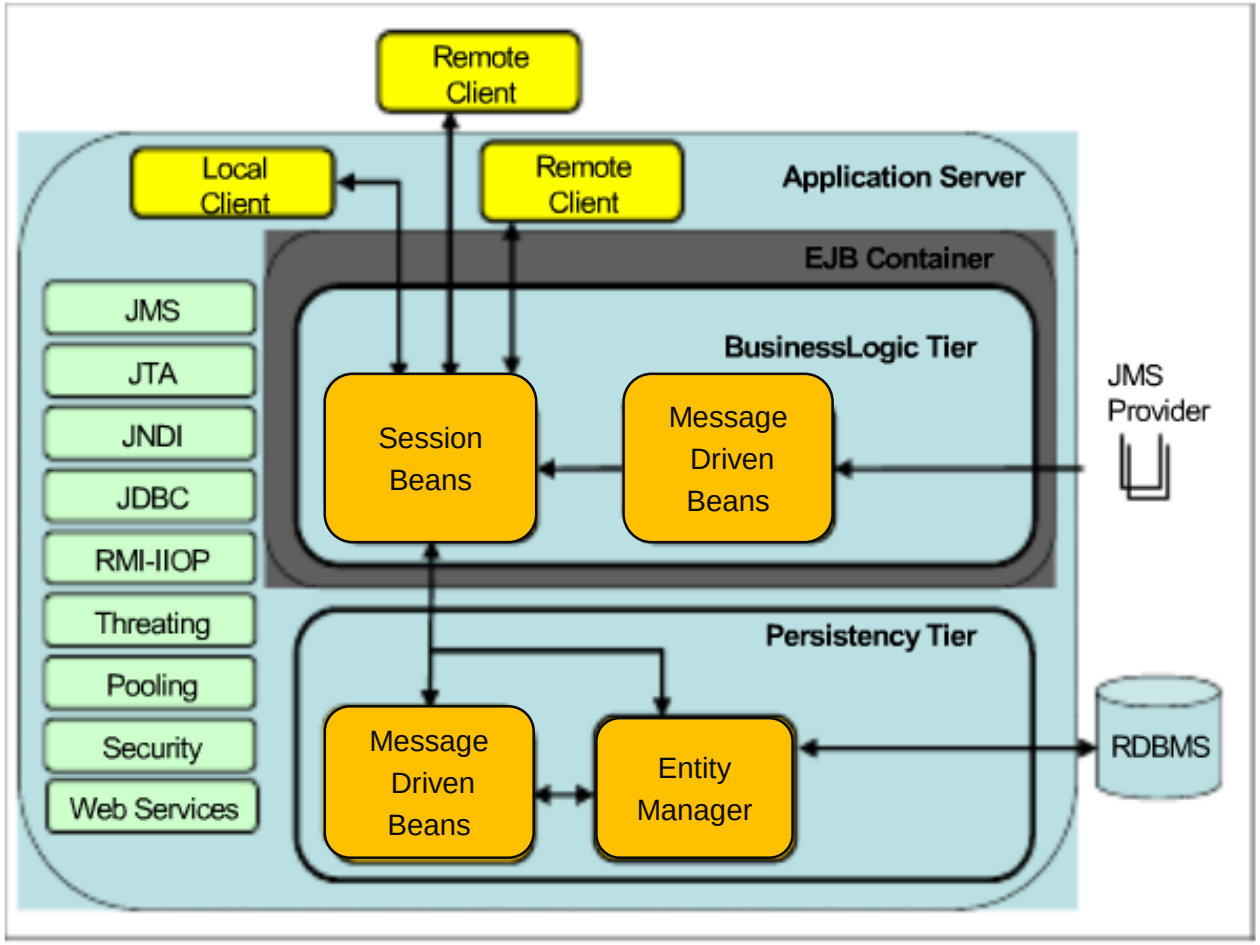
Funktionen des Application-Servers:

- Security
- Transaktions
- Directory Service
- Kommunikation zwischen Java EE Komponenten
- Persistenzdienst
- Livecycle Management der Komponenten
- Support for Deployment
- Kapselt OS Dienste und Resourcen

J2EE

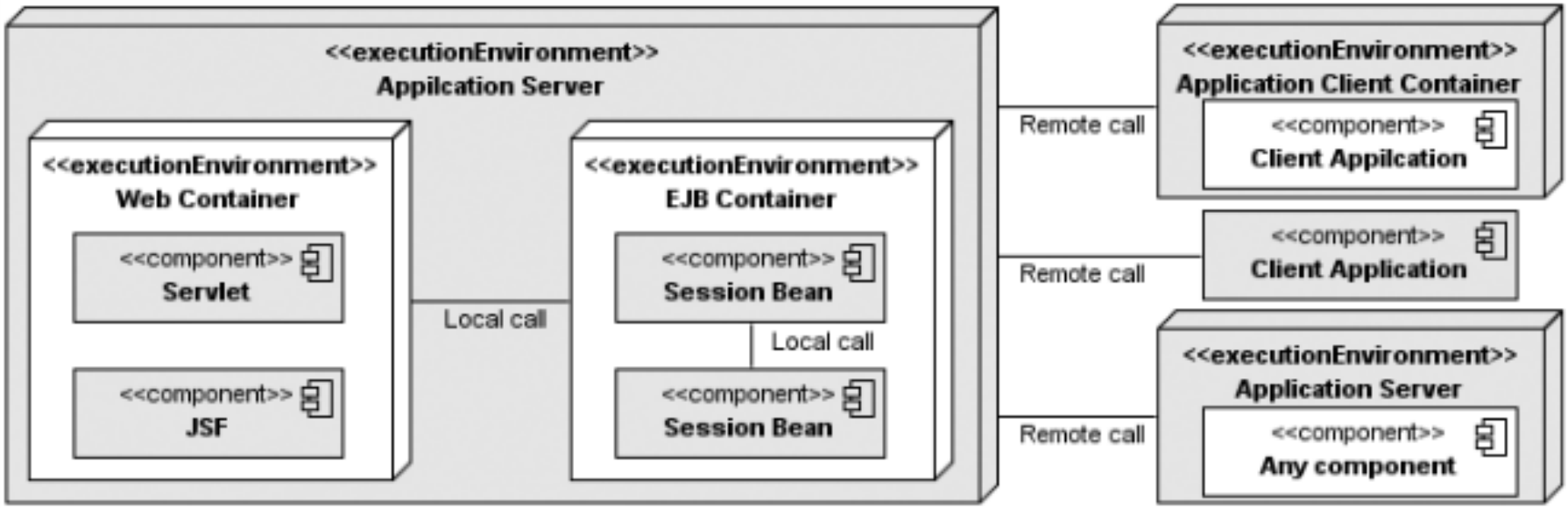

Presentation Layer
Business Layer
Database Layer
Client Layer
Web Layer
Business Logic Layer
Persistence Layer
Java EE 5 Container
Web Browser
Applet
Java Application Client
Application Client Container
Java Application Client
Web Container
Servlet
JSP page
JSF Component
EJB 3 Container
Session Bean
Message-Driven Bean
JPA Persistence Engine
Entity
Relational Database

J2EE



HWZ Hochschule für Wirtschaft Zürich

J2EE


<<executionEnvironment>>
Appilcation Server
<<executionEnvironment>>
Web Container
<<component>>
Servlet
<<component>>
JSF
Local call
<<executionEnvironment>>
EJB Container
<<component>>
Session Bean
Local call
<<component>>
Session Bean
Remote call
<<executionEnvironment>>
Application Client Container
<<component>>
Client Appilcation
Remote call
<<component>>
Client Application
Remote call
<<executionEnvironment>>
Application Server
<<component>>
Any component

Begriffe: J2EE

EJB	Enterprise Java Beans	beinhalten die Geschäftslogik einer Enterprise Anwendung oder gestatten Zugriff auf persistente Daten. Die Beans laufen in einem EJB-Container ab. Es gibt drei unterschiedliche Typen von EJBs: - Session-Beans, sowohl zustandsbehaftet als auch zustandslos, implementieren die Geschäftslogik und sind meistens vom Client zugreifbar - Message-Driven-Beans, kurz MDB, für die Verarbeitung von JMS-Nachrichten, wurden in Version 2.1 neu eingeführt - Entity-Beans für die Abbildung von persistenten Datenobjekten (ab Version 3.0 obsolet, da EJBs durch Detachment auch außerhalb des Containers nutzbar sind)
JSS	Java Servlet API	erlaubt im Allgemeinen die Erweiterung von Servern, deren Protokoll auf Anfragen und Antworten basiert. Primär werden Servlets im Zusammenhang mit dem Hypertext Transfer Protocol (HTTP) verwendet, wo sie in einem Web-Container leben und Anfragen von Webbrowsern beantworten.
JSP	JavaServer Pages	sind Textdokumente, die zum einen aus statischem Text und zum anderen aus dynamischen Textelementen – den JSP-Elementen – bestehen. Die JSP-Seiten werden transparent vom Web-Container in ein Servlet umgewandelt.
WS	Webservices	definieren Schnittstellen zu EJBs, die mit einem Uniform Resource Identifier (URI) eindeutig identifizierbar sind und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können.
JNDI	Java Naming and Directory Interface	ist eine gemeinsame Schnittstelle mit der alle Java-Klassen auf Namens- und Verzeichnisdienste zugreifen können. Über JNDI wird insbesondere der Zugriff auf Java-EE-Komponenten sichergestellt.
JMS	Java Message Service	ist eine API für die asynchrone Nachrichtenverarbeitung.
JTA	Java Transaction API	erlaubt der Anwendung die Steuerung der Transaktionsverwaltung. JTA ist die Java-Schnittstelle zu Transaktionsmonitoren. Standardmäßig wird diese Schnittstelle implementiert vom Java Transaction Service (JTS), welcher eine Schnittstelle zu CORBA Object Transaction Service (OTS) bietet.
JAAS	Java Authentication and Authorization Service	ist eine Java-API, die es ermöglicht, Dienste zur Authentifikation und Zugriffsrechte in Java-Programmen bereitzustellen. JAAS implementiert ein Standard-Pluggable Authentication Module (PAM) und unterstützt durch dieses Modul eine einfache Authentifizierung und benutzerbasierte Autorisierung.
	JavaMail	erlaubt den Zugriff auf Mail Services, wie z. B. SMTP, POP3 oder IMAP.
JAXB	Java Architecture for XML Binding	ermöglicht es, ein XML-Schema direkt an Java-Klassen zu binden. Wurde offiziell erst seit Java EE Version 1.5 gefordert, wird jedoch evt. schon vorher unterstützt.
JAXP	Java API for XML Processing	hilft dem Entwickler bei der Bearbeitung von XML-Dokumenten.
JAX-RPC	Java API for XML-based RPC	ermöglicht den entfernten Zugriff auf RPC-Dienste.
JAXR	Java API for XML Registries	dient dazu, einen transparenten Zugriff auf so genannte Business-Registries, wie beispielsweise ebXML oder ein UDDI basiertes Verzeichnis sicherzustellen.
JACC	Java Authorization Contract for Containers	definiert diverse Sicherheitsrichtlinien für die diversen Java-EE-Container.
JCA	J2EE Connector Architecture	dient dazu, andere Systeme transparent zu integrieren (Stichwort: EAI).
JAF	JavaBeans Activation Framework	bietet die Möglichkeit, verschiedene Daten anhand des MIME-Headers zu erkennen.
JAX-WS	Java API for XML Web Services	hilft bei der Erstellung von Web Services und zugehörigen Clients, die über XML kommunizieren, z. B. über SOAP.
	Web Service Metadata	beschreibt Web-Services mit Java-Annotationen
JPA	Java Persistence API	stellt eine einheitliche und datenbankunabhängige Schnittstelle für Object-Relational-Mapping und das Arbeiten mit Entitäten bereit.
StAX	Streaming API for XML	ist eine Cursor basierte XML-Verarbeitung in Ergänzung der DOM- und SAX-Parser
JSF	JavaServer Faces	dient dazu Komponenten für Benutzerschnittstellen in Webseiten einzubinden und die Navigation zu definieren.
JSTL	JavaServer Pages Standard Tag Library	ist eine Sammlung von JSP-Tags für die Strukturierung, XML, SQL, Internationalisierung usw.
CDI	Contexts and Dependency Injection	ist eine Technik um Felder nach dem Inversion of Control-Prinzip zu setzen. Es verbindet JSF mit EJB.

J2EE (Application-Servers)

Product	Vendor	Edition	Release date	Java EE compatibility [1]	Servlet spec	JSP spec
GlassFish	GlassFish Community	3.1.2.2	2012-07-17	6	3.0	2.2
WebSphere Application Server	IBM	8.5.0.0	2012-06-15	6	3.0	2.2
Interstage application server powered by Windows Azure	Fujitsu	1	2011-07-29	6	3.0	2.2
WebSphere AS Community Edition	IBM	3.0.0.3	2012-11-30	6	3.0	2.2
Geronimo	ASF	3.0.0	2012-07-13	6	3.0	2.2
Apache TomEE	ASF	1.0.0	2012-04-30	6	3.0	2.2
Interstage Application Server	Fujitsu	10.1	2011-12-27	6	3.0	2.2
JBoss AS	Red Hat	7.1.1.Final	2012-03-09	6	3.0	2.2
Siwpas	MechSoft	2.0.0	2011-07-29	see [2]	3.0	2.2
GlassFish Server	Oracle Corporation	9.1	2007-09-17	5	2.5	2.1
Geronimo	ASF	2.2.1	2010-12-11	5	2.5	2.1
WebLogic Server	Oracle Corporation BEA Systems	12c (12.1.1)	2011-12	6	3.0	2.2
Resin	Caucho Technology	4.0.24	2011-11	6 see [3]	3.0	2.2
JOnAS	OW2 Consortium (formerly ObjectWeb)	5.1.2	2010-05-04	5	2.5	2.1
JEUS	TmaxSoft	6	2007-06-07	5	2.5	2.1

Enterprise Java Beans (EJB)

EJB 3.0 Unterschied zu früher

EJB 3.0 ist viel Einfacher (Konzentration auf Business-Logic)

- Entity EJB sind POJO (Persisetnz wird auf Java Persistent API (JPA) gemacht)
- Kein Home-Interface nötig (Nur noch Business Interfaces)
- Specielle Interfaces und Deployment Descriptoren wurden durch Annotations ersetzt
- Intercepter Möglichkeiten ruft User-Functions (beim Aufruf von Business Functions or Life Cycle Events)
- Default Werte sind überall definiert (“configuration by exception” approach)
- Exception Handling; Notwendigkeit von checked exceptions wurde reduziert
- Meta-Daten werden durch Annotations beschrieben (Separation of concerns)

- ➔ Weniger drum herum

HWZ Hochschule für Wirtschaft Zürich

Servlet ruft Bean (im gleichen container) auf

```
import javax.ejb.Stateless;
@Stateless
public class HelloWorldBean {
      public String getHello() {
                return "Hello World !";
      }
}
```

Bean

HelloWorldBean.java

```
import java.io.*; import javax.ejb.EJB;
import javax.servlet.*; import javax.servlet.htteServlet;
public class TestServlet extends HttpServlet {
      @EJB
      private HelloWorldBean helloWorld;

      public void service (HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException {
                resp.getWriter().println(helloWorld.getHello());
      }
}
```

Servlet welches das Bean verwendet

TestServlet.java

HWZ Hochschule für Wirtschaft Zürich

EJB Java EE 5

```
import javax.ejb.Remote;
@Remote
public interface HelloWorld {
      String getHello();
}
```

Bean Interface

Nicht mehr nötig in EE 6

HelloWorld.java

```
import javax.ejb.Stateless;
@Stateless
public class HelloWorldBean implements HelloWorld {
       public String getHello() {
                  return "Hello World !";
       }
}
```

Bean

HelloWorldBean.java

```
import java.io.*; import javax.ejb.EJB;
import javax.servlet.*; import javax.servlet.htteServlet;
public class TestServlet extends HttpServlet {
       @EJB
       private HelloWorldBean helloWorld;

       public void service (HttpServletRequest req, HttpServletResponse resp)
                                        throws ServletException, IOException {
                                        resp.getWriter().println(helloWorld.getHello());
       }
}
```

Servlet welches das Bean verwendet

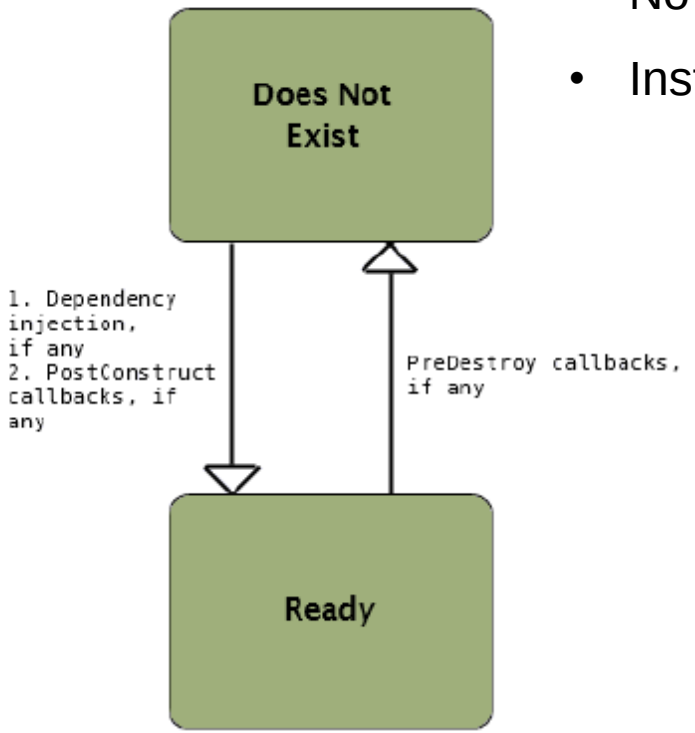
TestServlet.java

HWZ Hochschule für Wirtschaft Zürich

Stateless Session Beans

EJB: Stateless Session Beans

- Performs an operation for the client
- No associated conversational state
- May have instance state
- No concurrent access
- Instance variables do not keep the values across calls



States of a stateless session bean

EJB: Live cycle events Session Bean stateless

Note: The actual life cycle of a stateless session bean is independent of when a client obtains a reference to it. For example, the container might hand out a reference to the client, but not create the bean instance until some later time, for example, when a method is actually invoked on the reference. Or, the container might create a number of instances at startup time, and match them up with references at a later time.

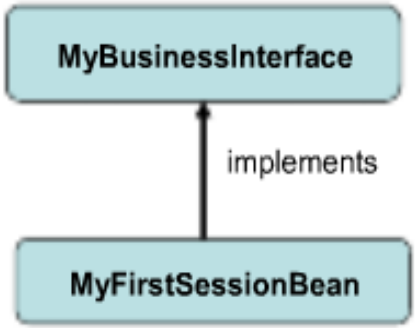
```
@Stateless
    :
    @PostConstruct
    public void initialize() {
        // initialize the resources uses by the bean
    }

    @PreDestroy
    public void cleanup() {
        // deallocates the resources uses by the bean
    }
    :
```

All stateless and stateful EJBs go through these two phases.

EJB 3.0: Session Bean (Stateless)

simply define a POJO

```
@Stateless

public class MyFirstSessionBean

                implements MyBusinessInterface {

// business methods according to
MyBusinessInterface

.....

}
```



- `MyFirstSessionBean` is a POJO that exposes a plain old Java interface (POJI), in this case `MyBusinessInterface`. This interface will be available to clients to invoke the EJB business methods.
- The **@Stateless** annotation indicates to the container that the given bean is **stateless session bean** so that the proper life cycle and runtime semantics can be enforced.
- **By default**, this session bean is accessed through a **local interface**. This is all you need to set up your first EJB! There are no special classes to extend and no interfaces to implement.

EJB 3.0: Session Bean (Stateless)

On the other hand, if we want to expose the same bean on the remote interface, we would use the **@Remote** annotation.

```
@Remote(MyRemoteBusinessInterface.class)
@Stateless
public class MyBean implements MyRemoteBusinessInterface {
// ejb methods
.....
}
```

Tipp: If the session bean implements only one interface, you can also just code **@Remote** (without a class name).

EJB: Session Bean (Business Interface)

Business interfaces

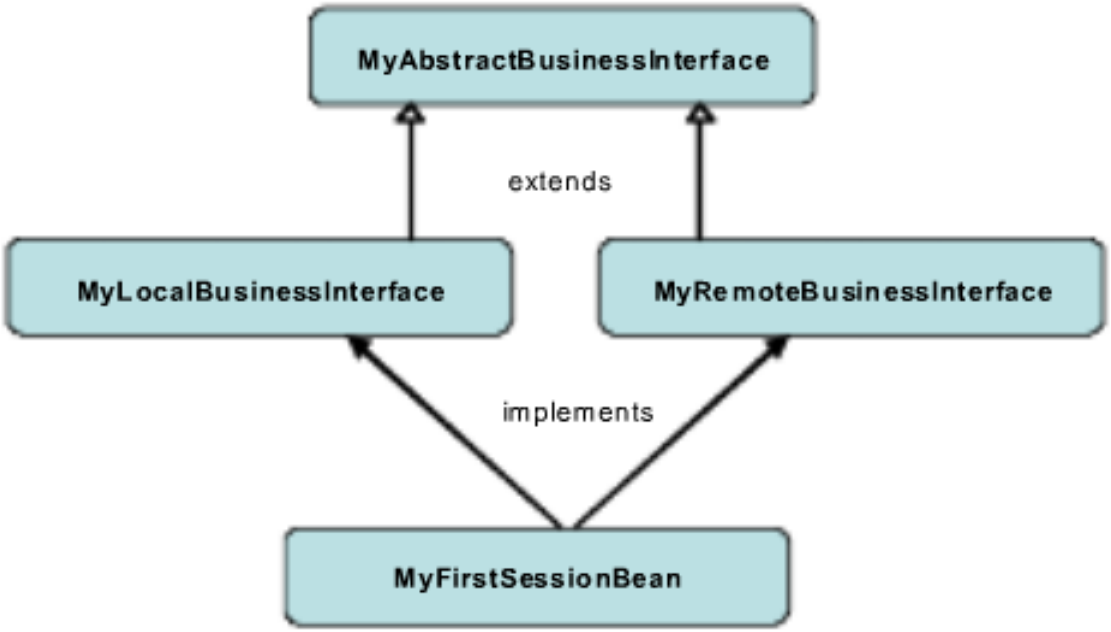
EJBs can expose different business interfaces, according to the fact that the EJB could be accessed either from a local or remote client.

We recommend that common behaviors to both local and remote interfaces should be placed in a super-interface, (See figure)

Consider the following aspects of business interfaces:

- A business interface **cannot** be both a **local** and a **remote** business interface of the bean.
- If a bean class implements a single interface, that interface is assumed to be the business interface of the bean. This business interface will be a local interface, unless the interface is designated as a remote business interface by use of the @Remote annotation or by means of the deployment descriptor.

This gives you a great flexibility during the design phase, because you can decide which methods are visible to local and remote clients.



EJB: Session Bean (Business Interface)

```
@Local
public interface MyLocalBusinessInterface extends MyAbstractBusinessInterface {
// methods declared in MyLocalBusinessInterface ......
}
```

```
@Remote
public interface MyRemoteBusinessInterface extends MyAbstractBusinessInterface {
// methods declared in MyRemoteBusinessInterface ......
}
```

```
@Stateless
public class MyFirstSessionBean implements MyLocalBusinessInterface, MyRemoteBusinessInterface {
// implementation of methods declared in MyLocalBusinessInterface
....
....
// implementation of methods declared in MyRemoteBusinessInterface
....
....
}
```

EJB: Session Bean (Business Interface)

```
@Stateless
@Local(MyLocalBusinessInterface.class)
@Remote(MyRemoteBusinessInterface.class)
public class MyFirstSessionBean implements MyLocalBusinessInterface,
                                           MyRemoteBusinessInterface {
// implementation of methods declared in MyLocalBusinessInterface
....
....
// implementation of methods declared in MyRemoteBusinessInterface
....
....
}
```

Another technique to define the business interfaces that are exposed, either as local or remote interfaces, is to specify **@Local** or **@Remote** annotations with the full class that implements these interfaces.

EJB: Session Bean (Business Interface Remote / Local)

```
public interface ItemLocal {
    List<Book> findBooks();
    List<CD> findCDs();
}

public interface ItemRemote {
    List<Book> findBooks();
    List<CD> findCDs();
    Book createBook(Book book);
    CD createCD(CD cd);
}
```

Usage

```
@Stateless
@Remote (ItemRemote)
@Local (ItemLocal)
public class ItemEJB implements ItemLocal, ItemRemote {
    ...
}
```

oder

```
@Local
public interface ItemLocal {
    List<Book> findBooks();
    List<CD> findCDs();
}

@Remote
public interface ItemRemote {
    List<Book> findBooks();
    List<CD> findCDs();
    Book createBook(Book book);
    CD createCD(CD cd);
}
```

Usage

```
@Stateless
public class ItemEJB implements ItemLocal, ItemRemote {
    ...
}
```

Stateful Session Beans

EJB: Live cycle events Session Bean stateful

Note: Because a stateful bean is bound to a particular client, it is best practice to correctly design session stateful beans to minimize their footprint inside the EJB container, and to correctly un-allocate it at the end of its life cycle, by invoking the method annotated with @Remove.

Stateful session beans have a timeout value. If the session stateful bean has not been used in the timeout period, it is marked inactive and is eligible for automatic deletion by the EJB container. Of course, it is still best practice for applications to remove the bean when the client is through with it, rather than relying on the timeout mechanism to do this.

```
@Stateful
    :
    @PostConstruct public void initialize()   { // initialze the resources uses by the bean }
    @PostActivate  public void postActivate() { // invoked just after activation }
    @PrePassivate  public void prePassivate() { // invoked just before passivation }
    @PreDestroy    public void cleanup()      { // deallocates the resources uses by the bean }
    @Remove        public void remove()       { // invoked explicitly by the client  }
    :
}
```

EJB 3.0: Session Bean (Stateful)

simply define a POJO

```
@Stateful
public class MySecondSessionBean
        implements MyBusinessStatefulInterface {
// ejb methods
.....
}
```

The **@Stateful** annotation indicates to the container that the given bean is a **stateful session bean** so that the proper life cycle and runtime semantics can be enforced.

Difference between Staeless and Staeful: http://www.java-samples.com/showtutorial.php?tutorialid=841

Singleton Session Beans

EJB: Singleton Session Beans

- Similar to stateful session beans but only one intance in the container.
- Can be used to share data between different components within a web-application independent of sessions
- ➔ Shared memory between components within an application

Übung: Session Bean

- Implementieren Sie ein **stateless** `CalculatorBean` (Neues Projekt: `HWZ_SessionBean`) das verschiedenen mathematische Berechnungs-Funktionen zur Verfügung stellt. Bevor sie Business-Methoden programmieren, implementieren Sie eine Methode mit der Annotation `@PostConstruct` resp `@PreDestroy` in der sie ein `System.out.println("CalculatorBean initiated!")` machen, ebenfalls machen Sie im Ctr dieser Bean eine `System.out.println("CalculatorBean Ctr called!")`. Im weiteren eine Methode `String ping()` welche den Klassen-Namen der Bean sowie deren Version (hard-codiert) zurückgibt. Machen Sie ein Deployment und schauen im Server log nach der entsprechenden Meldung.
- Schreiben Sie einen Java-Client , der die Ping-Funktion aufruft und den Werte anzeigt.
- Definieren Sie in `CalculatorBean` eine Variable `counter` und eine Methode `int incCounter()`. Rufen Sie mehrmals diese Methode von ihrem Client aus auf und schreiben den Wert auf die Console. Starten Sie ihren Client mehrmals nacheinander und beobachten diesen Wert. Was können Sie dazu aussagen?

Übung: Session Bean

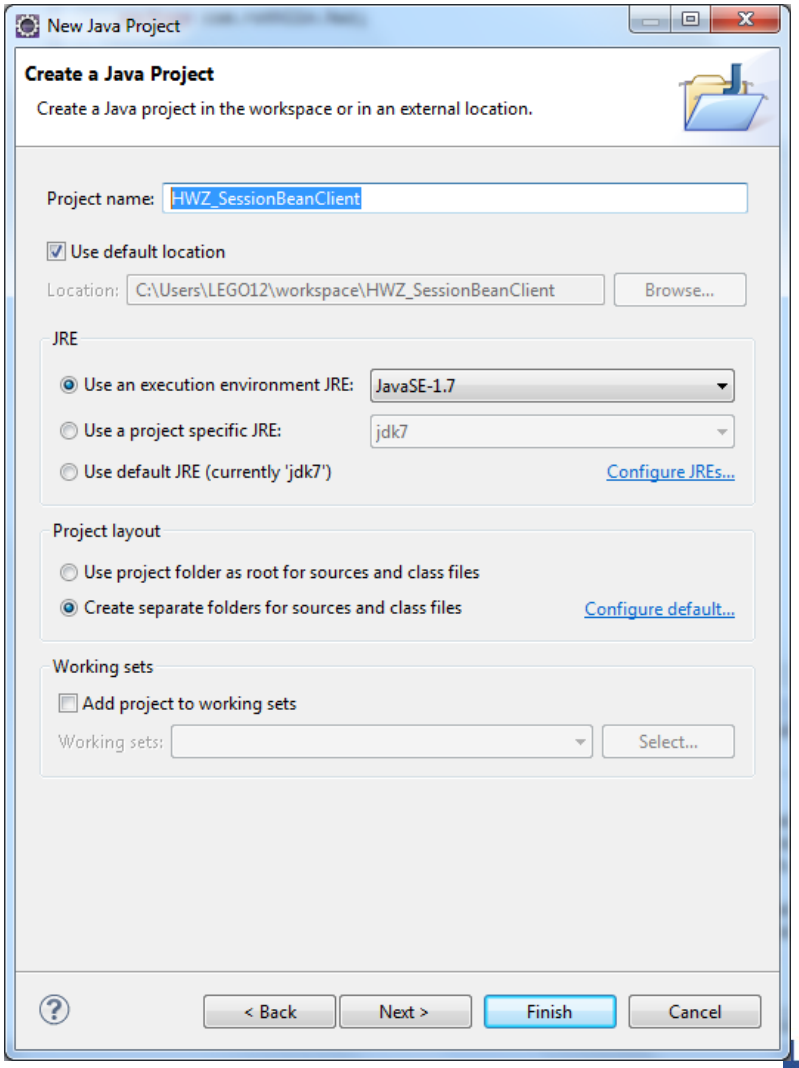
- Fügen sie nun Business-Methode z.B. `int multInt(int,int)`,`boolean isPrimefactor(int)` hinzu und rufen Sie diese von Ihrem Client aus auf.
- Implementieren Sie ein **singleton** `HitCounter_Singleton` bean (Im gleichen Projekt: `HWZ_SessionBean`) ebenfalls mit allen Methoden und Trace-Details von `CalculatorBean` wie `@PostConstruct`,`@PreDestroy`,`ping()` und den `System.out.println()`. Machen Sie ein Deployment und schauen im Server log nach den entsprechenden Meldung.
- Rufen Sie die entsprechenden Methoden aus Ihrem Java-Client aus auf.
- Fügen sie nun Business-Methode z.B. `String showCounterValue()`, `int incCounter()` und rufen Sie diese von Ihrem Client aus auf.

Übung: Stateless Bean

- Create a client to call `multInt`-Methode from `CalculatorBean`

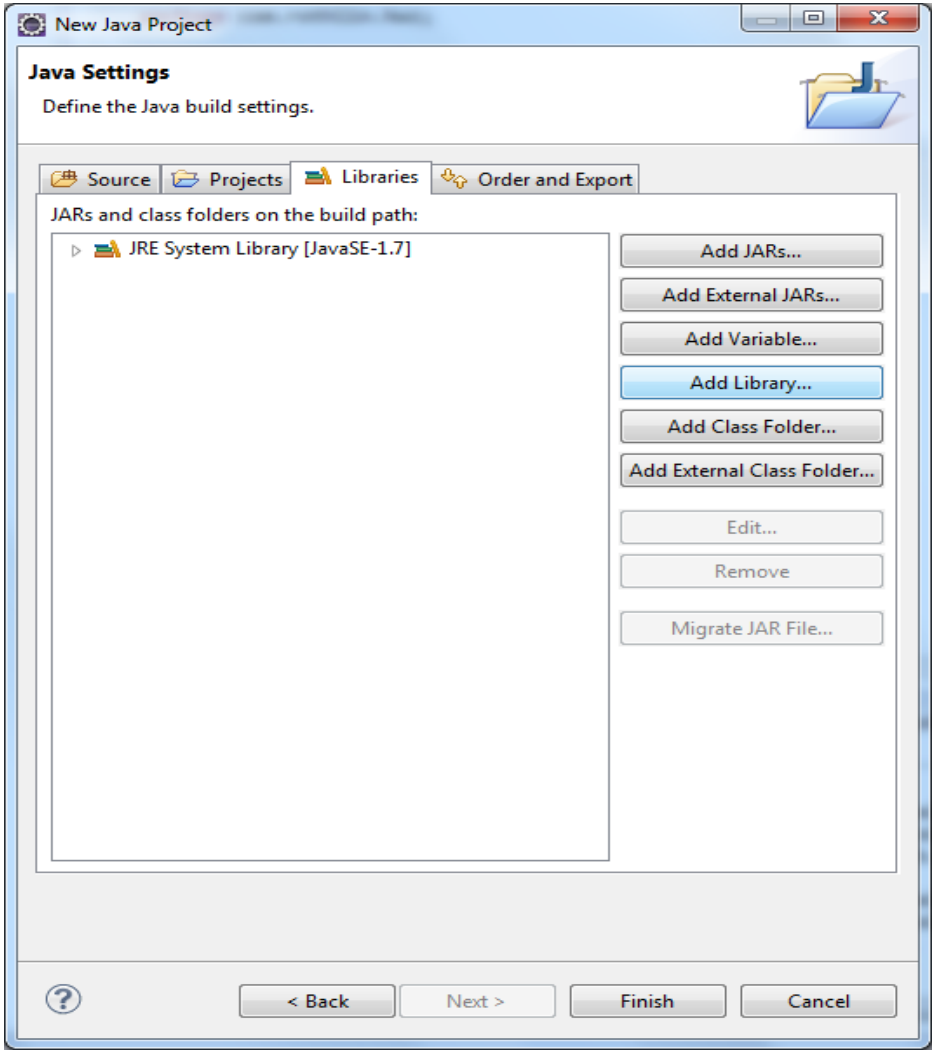
  ➔ Client is a POJO!

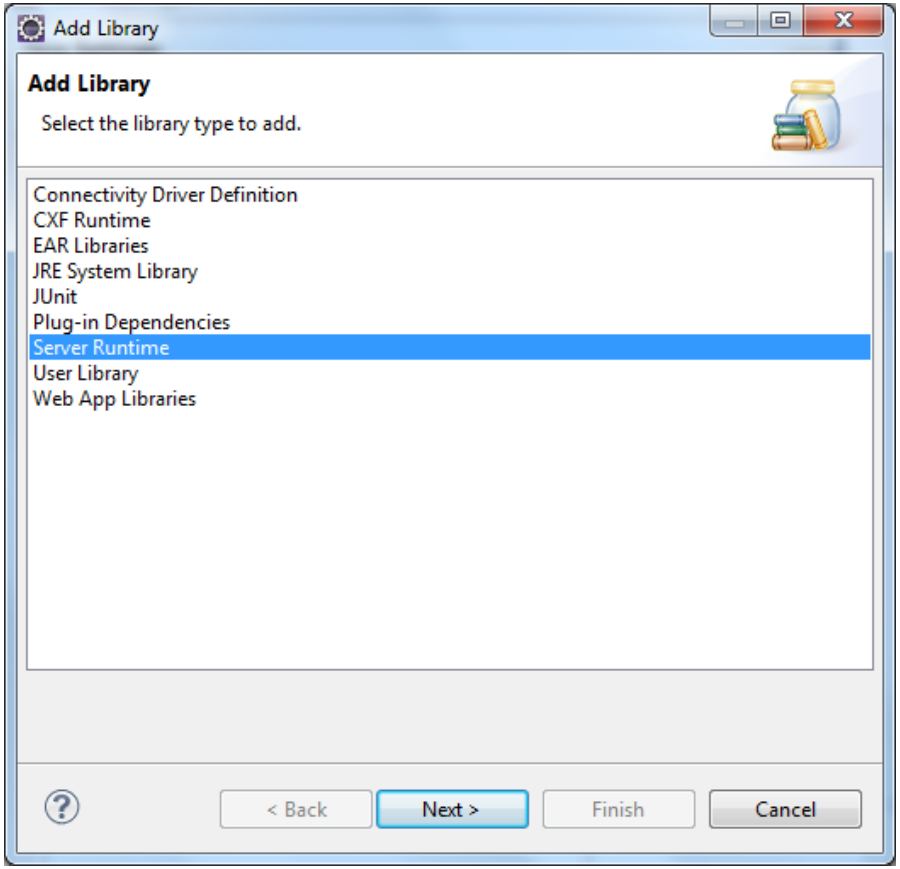
Create new Java Project
`HWZ_SessionBeanClient`



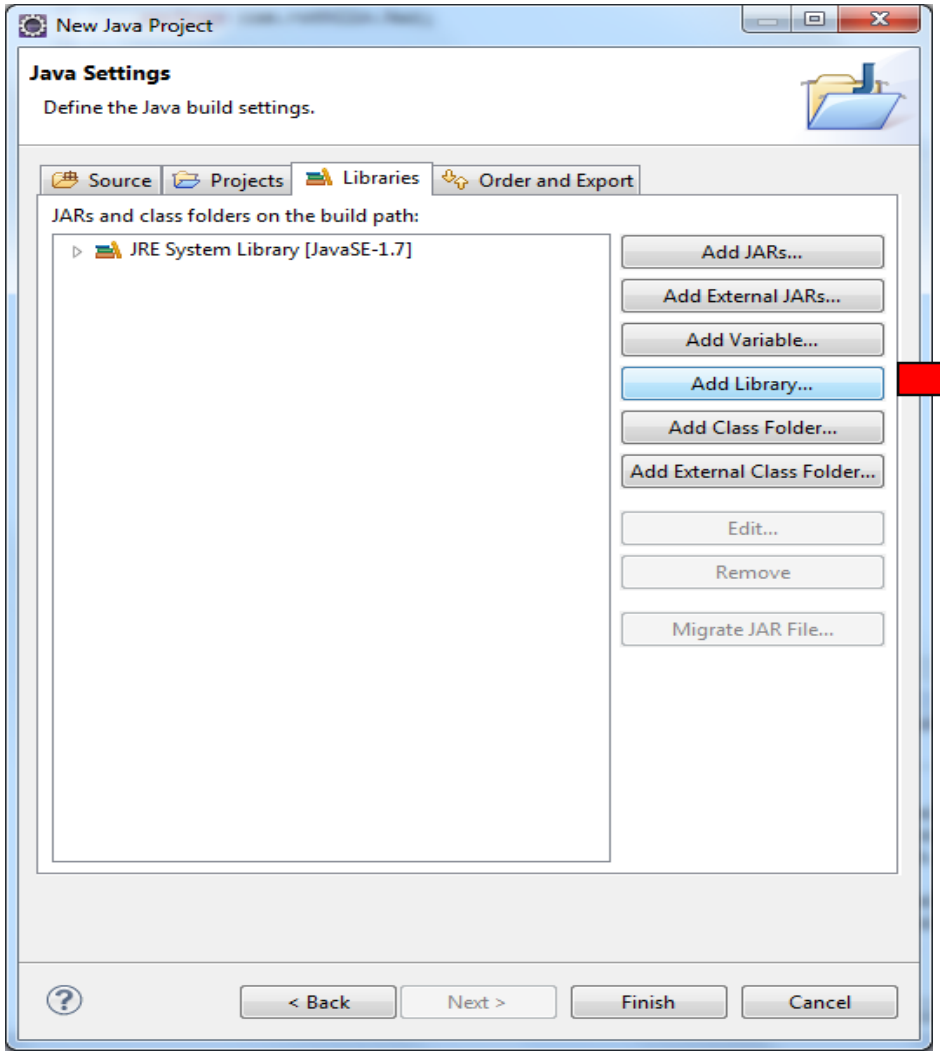
Übung: Stateless Bean

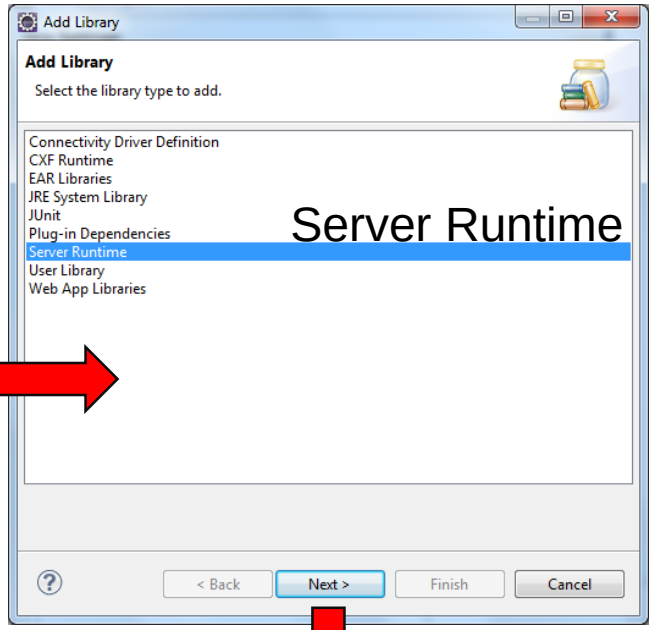
Add Libraries (Server Runtime)

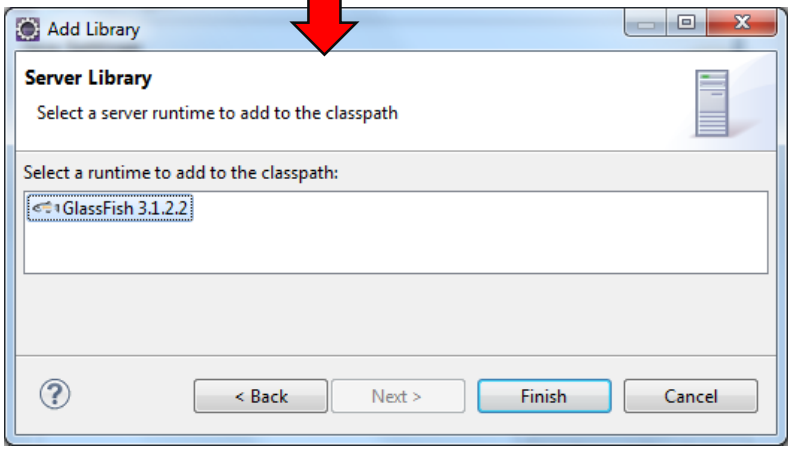




Übung: Stateless Bean

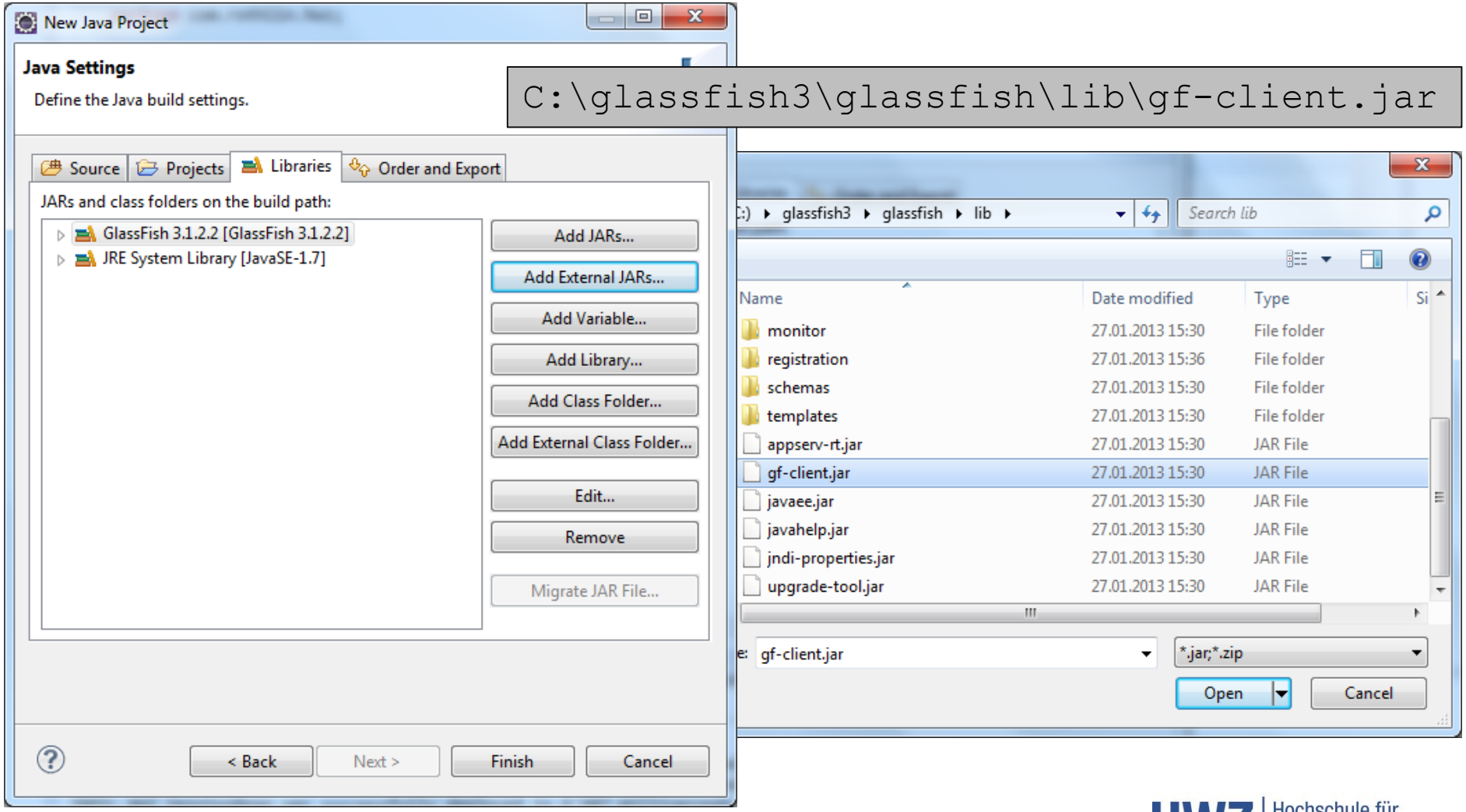




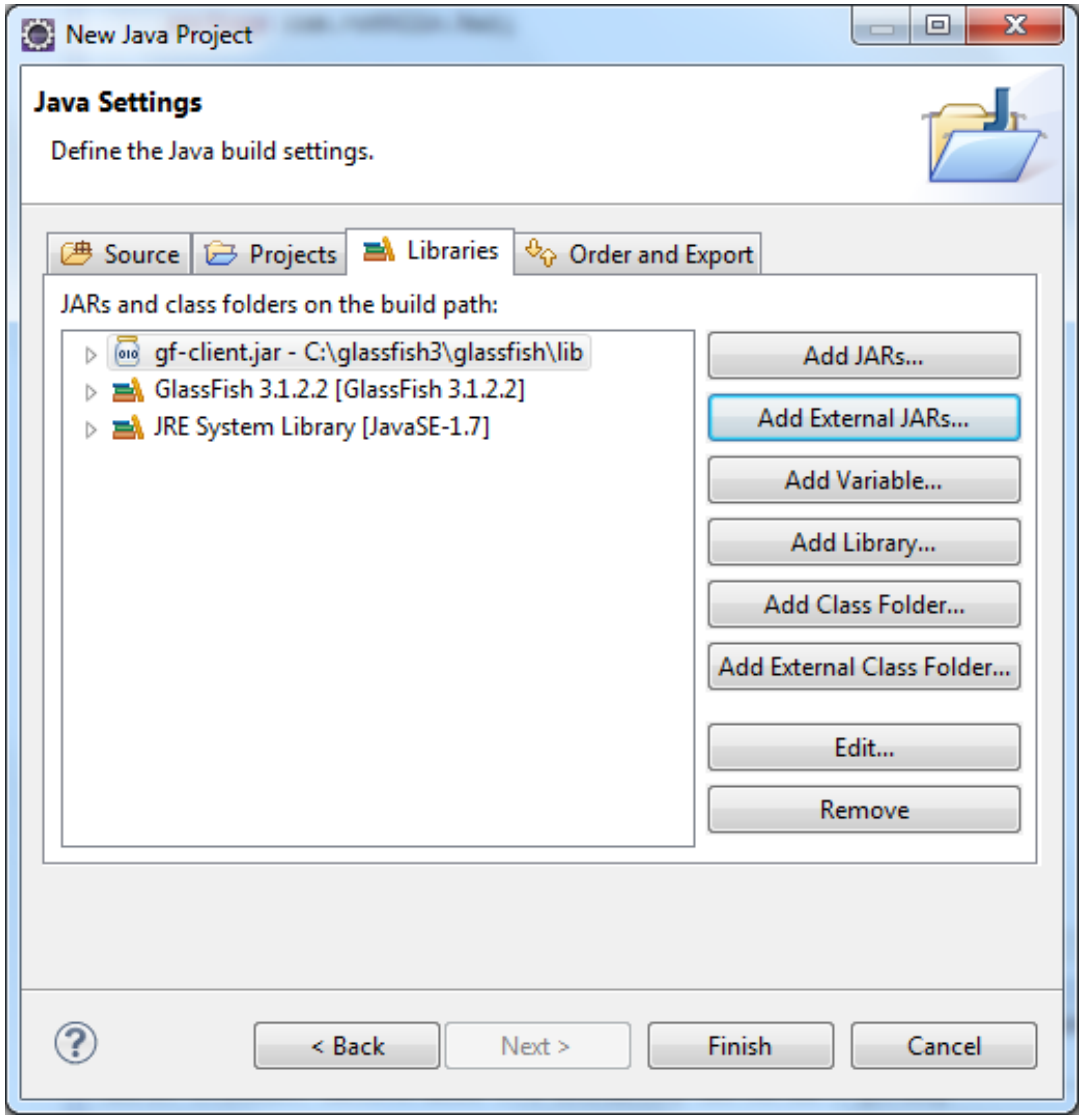


Übung: Stateless Bean

Add External JAR



Übung: Stateless Bean



Übung: Stateless Bean

```
package com.rothlin.hwz;
import javax.ejb.Stateless;
@Stateless
public class CalculatorBean implements CalculatorBeanRemote, CalculatorBeanLocal {
        public CalculatorBean() {}
        public int multInt(int a, int b) {
                return a * b;
        }
}
```

CalculatorBean.java

```
package com.rothlin.hwz;
import javax.ejb.Remote;
@Remote
public interface CalculatorBeanRemote {
  public int multInt(int a, int b);
}
```

CalculatorBeanRemote.java

```
package com.rothlin.hwz;
import javax.ejb.Remote;
@Local
public interface CalculatorBeanRemote {
  public int multInt(int a, int b);
}
```

CalculatorBeanLocal.java

Übung: Client to call Stateless Bean

```
import javax.naming.InitialContext;
public class BeanClientMain {
   public void runTest() throws Exception {
       System.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
       InitialContext ctx = new InitialContext();

       System.out.println("Calling Calc Bean (Session stateless)");
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
       ctx.addToEnvironment("org.omg.CORBA.ORBInitialPort", "3700");
       CalculatorBeanRemote bean = (CalculatorBeanRemote) ctx.lookup
              ("java:global/HWZ_SessionBean/CalculatorBean!com.rothlin.hwz.CalculatorBeanRemote");

        int result = bean.multInt(4, 5);
        System.out.println("4 * 5 = " + result);
   }

   public static void main(String[] args) {
         BeanClientMain cli = new BeanClientMain();
         try { cli.runTest(); }
         catch (Exception e) { e.printStackTrace(); }
   }
}
```

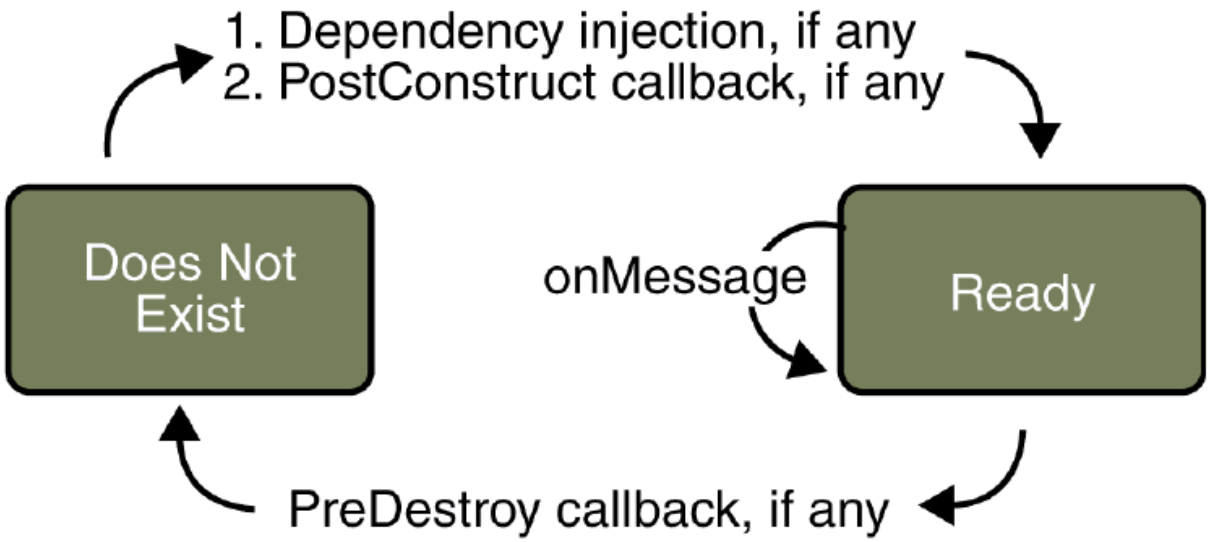
Nach dem Deployment des Beans auf den Application-Server findet man diesen String im Log-File

BeanClientMain.java

Message-Driven Beans

EJB: Message-Driven Bean (MDB)

- Retains no data or conversational state for a specific client
- All instances of an MDB are equivalent (pooling)
- A single MDB can process msg from multiple clients
- MDB are invoked asynchronously
- MDB are stateless

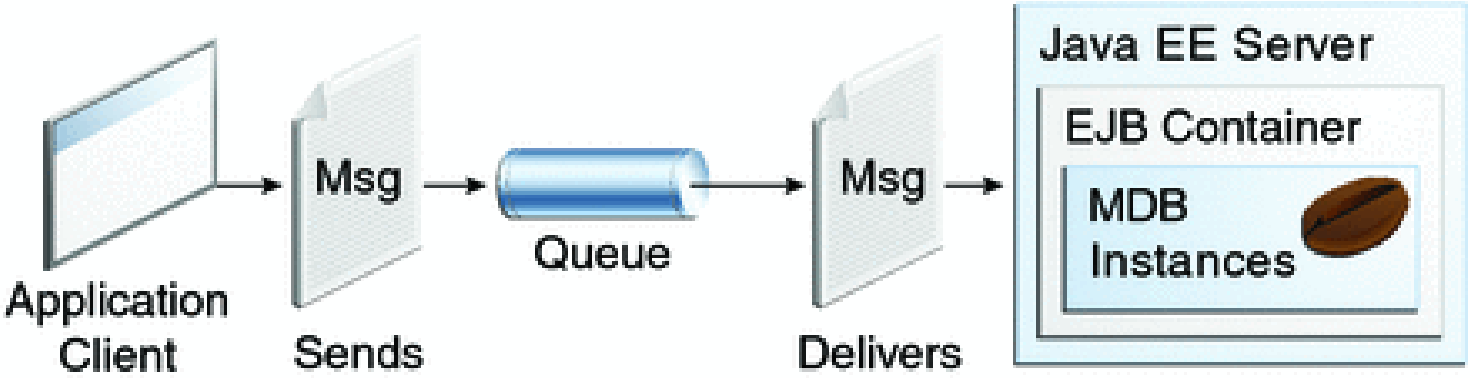


States of a message-driven bean

EJB: Message-Driven Bean Example

`SimpleMessageClient`: An application client that sends several messages to a queue

`SimpleMessageBean`: A message-driven bean that asynchronously receives and processes the messages that are sent to the queue



JMS (Java Message Service)

Point-To-Point: Sender schreibt msg in eine Queue und **ein** Empfänger kann diese empfangen

Publish-Subscribe: Sender broadcasted Meldungen unter einem **Topic** (Domain). Mehrere Empfänger können sich auf diese Topic abonnieren und erhalten diese Meldungen

MDB Beispiel: http://www.roseindia.net/ejb/example-of-messageBean.shtml

EJB: Message-Driven Bean

In this way, they can be thought of as another interaction mechanism for invoking EJBs, but unlike session beans, the container is responsible for invoking them when a message is received, not a client (or another bean).

To define a message-driven bean in EJB 3.0, you declare a POJO.

```
@MessageDriven(activationConfig = {
        @ActivationConfigProperty(
                propertyName="destinationType",
                propertyValue="javax.jms.Queue"),
        @ActivationConfigProperty(
                propertyName="destination",
                propertyValue="queue/myQueue")
})
public class MyMessageBean implements javax.jms.MessageListener {
        public void onMessage(javax.msg.Message inMsg) {
                // implement the onMessage method
                // to handle the incoming message
                ....
        }
}
```

EJB: Message-Driven Bean

In EJB 3.0, the MDB bean class is annotated with the **@MessageDriven** annotation, which specifies a set of activation configuration parameters.

- These parameters are unique to the particular kind of JCA (**Java EE Connector Architecture** not Java Cryptography Architecture) adapter that is used to drive the MDB. Some adapters have configuration parameters that let you specify the destination queue of the MDB. In the case where the adapter does not support this, the destination name must be specified using a <message-destination> entry in the XML binding file.
- The bean class has to implement the **MessageListener** interface, which defines only one method, **onMessage**. When a message arrives in the queue monitored by this MDB, the container calls the onMessage method of the bean class and passes the incoming message in as the parameter.
- Furthermore, the **ActivationConfigProperty** of the **@MessageDriven** annotation provides messaging system–specific configuration information.

EJB: Best Practices

An EJB 3.0 developer must adhere to the following basic rules:

- Each session bean must be a POJO, the class must be concrete (therefore neither abstract or final), and must have a no-argument constructor (if not present, the compiler will insert a default constructor).
- The POJO must implement at least one POJI. We stress at least, because you can have different interfaces for local and remote clients.
- If the business interface is @Remote annotated, all the values passed through the interface must implement java.io.Serializable. Typically the declared parameters are defined serializable, but this is not required as long as the actual values passed are serializable.
- A session EJB can subclass a POJO, but **cannot** subclass another session EJB.

Persistenz / JPA (Entity Beans)

Übung: Persistenz

- Implementieren Sie eine **JSP / Servlet**, welche die Parameter **Name** und **Age** in einer mySql DB ablegt. Die DB connection wird über eine JNDI-Resource definiert.
  `(HWZ_JSP_DB_Connection:index.jsp)`

Anhang

Youtube Tutorials

http://www.youtube.com/watch?v=Dz86V4brPUE	1.1 Einführung in Java EE 6
http://www.youtube.com/watch?v=xaGYtuq-8aE&feature=relmfu	1.2 Einführung in's Beispiel (Spiele shop)
http://www.youtube.com/watch?v=i8EoeWwxzmc&feature=relmfu	2.1 Set-up von Eclipse, JRE
http://www.youtube.com/watch?NR=1&v=OZgv8WtWQRI&feature=endscreen	2.2 Set-Up JDK, Glassfish Plug-in
http://www.youtube.com/watch?v=EOdvmGVdndA	2.21 Login JSP / Servlet
http://www.youtube.com/watch?v=UFHL2S1cibY&NR=1&feature=endscreen	2.3 MySql-Setup: XAMPP, DB conncetion aus Eclipse raus
http://www.youtube.com/watch?v=d0D_TsEGes4	2.4 Einrichten DB Connection Pool in Glassfish (inkl. JNDI Name)
Definition)	
http://www.youtube.com/watch?v=Cegnb1X8zrk&feature=relmfu	2.5 Erste Web-application (JSP mit DB Zugriff)
http://www.youtube.com/watch?v=G6UgxGXqPtU&feature=relmfu	2.6 DB Design mit Eclipse Plug-In
http://www.youtube.com/watch?v=g7jddsObpbM	4.1 Create stateless EJB
http://www.youtube.com/watch?v=B8j4U_do-nA	4.2 Deploy stateless EJB
http://www.youtube.com/watch?v=fXK0p3QhFLQ&feature=BFa&list=PLF124E0D4576E2187	5.1 Create and Deploy statefull EJB
http://www.youtube.com/watch?v=TtXmDgUx8nw&feature=BFa&list=PLX6QZhqJAMDClY5ODAtFnEj0wZD3-fsib	5.2 Build a client to call EJB
http://www.youtube.com/watch?v=TtXmDgUx8nw	5.3 Create and run client

Youtube Tutorials (Cont.)

EJB 3.0 (YouTube: ejb 3.0 tutorial in eclipse)

http://www.youtube.com/watch?v=W-Tq93LUdMM	Kurze Einfuehrung
http://www.youtube.com/watch?v=JrDT-PYhG20	Session Bean
http://www.youtube.com/watch?v=g7jddsObpbM	Session Bean with Eclipse and glassfish
http://www.youtube.com/watch?v=TtXmDgUx8nw&feature=BFa&list=PLX6QZhqJAMDClY5ODAtFnEj0wZD3-fsib	Calling / Using of EJB

http://www.youtube.com/watch?v=9Kf5m7bMu74	6.1 Set-Up Eclipse with Glassfish
http://www.youtube.com/watch?v=ppGqtOeHm-g	6.2 JSP / Servlet calling an EJB
http://www.youtube.com/watch?v=32wpTh1TmY4	6.3 JPA
http://www.youtube.com/watch?v=RzWK-DXWZ8c	6.4 JFS
http://www.youtube.com/watch?v=kfbgeLGAi3Q	6.5 REST Web-Service

Nicht gebrauchte Folien